TURING 图灵程序设计丛书

WILEY

[美] Jake Rutter 著
魏忠 译

Smashing jQuery

精彩绝伦的jQuery



人民邮电出版社
POSTS & TELECOM PRESS

图灵程序设计丛书

[美] Jake Rutter 著
魏忠 译

Smashing jQuery

精彩绝伦的jQuery

人民邮电出版社
北京

图书在版编目（CIP）数据

精彩绝伦的jQuery /（美）拉特（Rutter,J.）著；
魏忠译. -- 北京：人民邮电出版社，2012.5
（图灵程序设计丛书）
书名原文：Smashing jQuery
ISBN 978-7-115-28065-7

Ⅰ. ①精… Ⅱ. ①拉… ②魏… Ⅲ. ①JAVA语言—程
序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2012)第081887号

内 容 提 要

本书是 jQuery 基础教程，通过大量实用技巧、案例、示例分 4 部分全面讲解了 jQuery 开发。第一部分介绍 jQuery 与 JavaScript 库，以及 jQuery 带来的巨大便利。第二部分重点论述 jQuery 基础知识，包括选择器、事件与特效。第三部分探讨 jQuery 应用，包括用 jQuery 改进 Web 表单验证等内容。第四部分是 jQuery 高级技术分析，如使用和编写插件、处理 Ajax 请求、编写移动应用。另外，本书最后盘点了 jQuery 线上资源。

本书适合 Web 设计人员及前端开发人员学习参考。

图灵程序设计丛书
精彩绝伦的jQuery

◆ 著　[美] Jake Rutter
译　魏　忠
责任编辑　毛倩倩
◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街14号
邮编　100061　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京　印刷
◆ 开本：800×1000　1/16
印张：18.25　彩插 4
字数：430千字　2012年 5 月第 1 版
印数：1 – 3 500册　2012年 5 月北京第 1 次印刷
著作权合同登记号　图字：01-2011-2968号
ISBN 978-7-115-28065-7

定价：59.00元
读者服务热线：(010)51095186转604　印装质量热线：(010)67129223
反盗版热线：(010)67171154

版 权 声 明

前　言

在我每日的Web设计与开发工作中，jQuery是必不可少的一部分。Wiley出版社联络到我，希望我写一本关于jQuery的书，这让我既激动又惶恐。jQuery让Web设计有望走进一个无所不能的新世界。我希望能通过这本书，告诉人们如何使用jQuery提高开发效率，以及如何编写人机交互组件，而后者人们普遍认为没有高深的编程知识就不可能做到。

本书既是jQuery入门读物，又是一本jQuery实用指南，书中的例子都是实际Web开发过程中的解决方案，因此读者完全可以将本书作为日常开发工作的手边书。本书内容分为4部分，下面将分别介绍。

第一部分　jQuery 与 JavaScript 简介

本部分主要是向初学者介绍jQuery。这一部分介绍了JavaScript库，以及这些库如何一步步走进Web设计师和开发者的“工具箱”，并逐渐成为其中的重要角色。其中重点阐释了jQuery带来的巨大便利，让你彻底搞清楚jQuery如此流行的原因。本部分还谈及了渐进增强技术的重要性。在弄清楚为什么要使用jQuery之后，我们马上开始学习使用jQuery并将它应用到Web站点。

第二部分　jQuery 基础

本部分带你一步步全面了解jQuery的基础知识，比如使用选择器、事件及特效等。jQuery的选择器极其强悍，因此我专门拿出整整一章的篇幅详细介绍各种选择器，不但介绍用法，还给出了例子。本部分还介绍了事件和特效，学好它们，你就为创建Web应用和UI组件打好了坚实基础。

第三部分　jQuery 应用

第三部分的主题是把前面学到的jQuery知识应用于具体的Web站点或应用程序。（第二部分有零零散散的例子，但没有完整的解决方案，比如如何创建一个折叠菜单或标签式导航等。）本部分还包括使用jQuery改进Web站点表单验证的内容。

第四部分　jQuery 高级技术

每位jQuery开发者都希望了解jQuery高级技术，如使用和编写插件、用jQuery处理Ajax请求，或者用jQuery编写移动应用。第四部分深入研究了这些高级主题。此外，我还专门拿出第12章盘点了所有的jQuery线上资源。

读者对象

本书面向那些刚刚开始使用jQuery的Web设计师和前端开发人员。可能你已经知道jQuery插件怎么安装，但还不清楚怎么写，或者在网上常常听人说起jQuery多么了不起，正想学习jQuery以改进自己的站点，那么请阅读本书吧。值得提一下的是，本书的读者最好熟悉HTML、CSS知识并了解基本的JavaScript知识。

关于本书

初次出现的术语会印刷成楷体，需要你键入的内容使用粗体。书中全部例子使用Firefox浏览器呈现，也兼容IE 6+、Firefox 2.0+、Safari 3.0+、Opera 9.0+及Chrome浏览器。若需下载书中的示例代码，请访问www.wiley.com/go/smashingjquery[①]。

① 示例代码也可在图灵社区（ituring. com. cn）本书网页免费注册下载。——编者注

致　　谢

感谢本书的策划团队：Chris Webb，是他给予机会，我才有幸写了这本有意思的书；Linda Morris，本书的项目编辑，帮我纠正了许多低级错误；Andrew Croxall和Dennis Cohen，作为技术编辑确保了书中代码和内容的正确性。

当然，还要感谢我的妻子，在编写本书的过程中，她给了我莫大的支持和帮助。如果没有她，我不可能胜任这样的挑战。感谢我的父母，多谢他们教导我努力工作就有回报，特别是在我尝试一件看上去不可能完成的事情时。感谢我现在及以前的老板，正是他们给了我参与jQuery这样的开源项目并挑战Web局限的机会。

最最感谢John Resig、jQuery团队和社区，他们创造的jQuery如此精彩绝伦，为我带来了数不清的工作机会，也给了我使用寥寥几行代码创建卓越Web应用的能力。

目　　录

第三部分 jQuery 应用

Part 1

第一部分

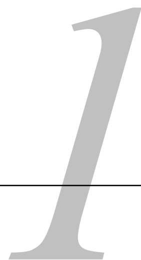
jQuery 与 JavaScript 简介

本部分内容

- 第 1 章　认识 jQuery
- 第 2 章　jQuery 入门

第 1 章

认识jQuery



jQuery是一个JavaScript库，它通过封装原生JavaScript函数得到了一整套定义好的方法。这些方法能有效地帮助Web设计师和开发者快捷编写和扩展JavaScript交互组件。jQuery没有提供任何新的功能，它最大的贡献是把JavaScript难懂难用的API整理成了易懂易用的jQuery语法，从而赢得了无数的用户。

在本章中，我们一起了解JavaScript库带来了哪些便利，有哪些与jQuery近似的库，并深入了解jQuery的特性，搞清是哪些因素造就了伟大的jQuery。

1.1 探索 JavaScript 库

构建在原生JavaScript指令上的常用函数构成了库，库使得Web设计师和开发者能方便地增强页面的交互能力及可用性。

我们可以把库当成一种框架或者蓝图，指导构建Web站点的一套规则或者指南。对设计师和开发者来说，使用JavaScript库写代码更容易——库是一个起点。有很多流行的库，如Prototype、MooTools、Dojo、YUI，当然还有本书的主角jQuery，它们被广泛应用于网络的每个角落。不同的库侧重点有所不同，jQuery擅长操作维护DOM（Document Object Model，文档对象模型）。

DOM实际上把代表Web页面的HTML代码表示成一个树形结构，其中每个枝杈都是属于一个层次结构的、彼此连接的一个节点。绝大多数情况下都通过CSS访问这些节点（设置样式），或借助选择器通过JavaScript访问它们。HTML标准委员会制定了维护HTML Web页面的API（Application Programming Interface，应用编程接口），也就是DOM，供Web设计师和开发者使用。HTML5为DOM补充了一些新的API，这些API能为因特网提供更好的用户体验。当网页完全加载之后，DOM就准备好和用户进行交互了。

通过在页面中包含一个库文件，设计师和开发者就可利用库中提供的特制方法扩展DOM。

1.1.1 JavaScript库优于传统解决方案之处

使用JavaScript库最大的好处是能避开那些乏味的非交互内容，利用库提供的大量方法扩展Web页面。

JavaScript库让Web设计师和开发者能够更方便地处理特效、动画、事件、Ajax及用户交互组件，从而提高开发效率。Web设计师和开发者不必局限于库提供的功能，完全可以自己实现需要的功能。

对那些了解DOM的Web设计师来说，与使用原生JavaScript的有限API相比，使用JavaScript库操作DOM更加简单。

如果没有JavaScript库，使用原生JavaScript实现同样的功能，那么我们将不得不花费数个小时，度过漫漫长夜，苦苦编程、测试、捉“虫”，最后写出长得吓人的代码。JavaScript库能有效、大幅地减少代码数量，比如实现同一目标时，若使用原生JavaScript需要写400行代码，而使用库的话则可能只需要写100行甚至更少。

使用JavaScript库的另一个好处是可以避免重复代码。因为要写一些JavaScript函数来完成相似的任务时，我们往往最终收获许多相似的代码，但如果用上了JavaScript库，你就能消除绝大部分的重复代码。

1.1.2　主流JavaScript库

目前，大约有20个（目前开发活跃的）JavaScript库，其中有5个库最流行，它们是YUI、Prototype、MooTools、Dojo和本书的主角jQuery。它们之所以脱颖而出，是因为很好用，并且都有着巨大的用户群。绝大多数库之间的差异主要在于库的体积及浏览器支持程度不同。

我即将探讨的5个库都是开源项目，这意味着每个人都可以为这些库贡献源代码。微软的软件是不开源的，是专属于微软公司的软件。微软公司雇用程序员开发软件，然后销售这些软件并收取授权费。交纳授权费之后，软件用户通常在一定期限之内有权使用这些软件，并且可在遇到麻烦时从微软公司得到帮助。

开源软件与之不同。任何人都能够下载源代码并对其进行改进，这样会造就更好的代码。因为所有的代码都是由志愿者写就的，而所有志愿者拥有同一个目标——写出更好的软件（而不是赚钱）。由于不必支付任何费用，这些库尽可随意使用。网上的开源社区极为庞大，有数百万用户通过博客或论坛贡献内容，而今Web设计师和开发者在遇到问题时可非常容易地得到支持。

在学习JavaScript库的过程中要牢记一点，你正在学习的库就像一种新语言——没错，它确实是JavaScript语言的另一种演绎。

1. YUI

YUI（Yhaoo! User Interface，雅虎用户界面）JavaScript库由雅虎开发者网络于2005年发布，它采用的是BSD许可证。BSD许可证允许以极其自由的方式传播软件，与其他类似许可证（如GNU GPL）相比，BSD许可证对软件传播的限制最少。YUI完全兼容IE 6+、Firefox 3+、Safari 3+以及Opera 10+。

YUI库文件的总大小约31 KB。

为了让你了解一下YUI代码是个什么样子，下面列出了一段JavaScript代码，它演示了如何使用YUI库实现`click`事件。这段代码中的`click`事件分为两部分：一个是`click`事件发生时被调用

的函数，一个是click事件本身。这些代码因为使用了YUI专用语法显得不那么优雅。

```
function handleClick(e) {
  Y.log(e);
}
YUI().use('node-base', function(Y) {
  Y.on("click", handleClick, "#foo");
});
```

2. Prototype

Prototype库由Sam Stevenson创建。由于是和非常流行的Web快速开发框架Ruby on Rails绑定发布的第一个JS框架，它很快流行起来。由于它是Ruby on Rails的一部分，我总感觉它并不适合Web设计师，而是更适合专业Web开发者结合Ruby on Rails使用。

Prototype库是一个包含Ajax功能的基础库，随着它的辅助库Scriptaculous的加入，Prototype库的功能越来越丰富。Scriptaculous负责提供特效及用户界面元素，是一个只能与Prototype一起使用的库。Prototype库的主要缺点在于尺寸：所有JavaScript文件加起来大约有278 KB。

对没有多少经验的前端开发者来说，Prototype和Scriptaculous库的文档是相当难以理解的。和其他库一样，Prototype也有一个技术支持社区。不过由于它的语法相当复杂，Prototype终究是一个难以学习的库。为了让你感受一下Prototype代码，下面的代码演示了如何使用Prototype库处理click事件。这里的click事件用法看上去与jQuery中的非常类似，别让表象欺骗了你——Prototype中许多的其他方法代码要比这个复杂得多，而且看起来不太像jQuery。

```
$("foo").observe("click", function() {
  alert('Clicked!');
});
```

3. MooTools

MooTools库首次发布于2006年，与Prototype有相似之处——语法（相对复杂）适合中高级Web设计人员和开发人员。MooTools是一个面向对象的框架，它以面向对象的方式增强了JavaScript API，也为Web页面提供一些人机交互功能。MooTools适合那些喜欢纯净JavaScript的人。

下面是使用MooTools库处理click事件的代码示例：

```
$('foo').addEvent('click', function() {});
```

4. Dojo

Dojo最早的版本发布于2004年，它的设计目标是创建兼容各种浏览器的Web应用，为站点平滑地添加交互功能。Dojo的语法相当复杂，给人的感觉就是在写原生JavaScript，它针对的用户群是那些有经验的前端开发人员，它的用法和思想不太适合初学者。

下面是使用Dojo库处理click事件的代码示例：

```
fooNode = dojo.byId("foo");
fooConnections = [];
fooConnections.push(dojo.connect(fooNode, 'onclick', foo));
```

看看前面这些例子，这些JavaScript库的语法相当吓人。现在我们看看jQuery怎么处理click

事件：

```
$('#foo').click(function() {
  //单击事件
});
```

1.1.3　jQuery的高明之处

jQuery天生“聪敏过人”。比较一下前面的代码示例，显然jQuery版本最简洁，也最容易理解。这是jQuery的高明之处——简单直接。没有多余的雕饰，没有费解的代码，写jQuery不需要专业的后端编程知识，不过这并不是说jQuery只能干点简单的小活。

图1-1展示的是jQuery官方网站http://jquery.com。

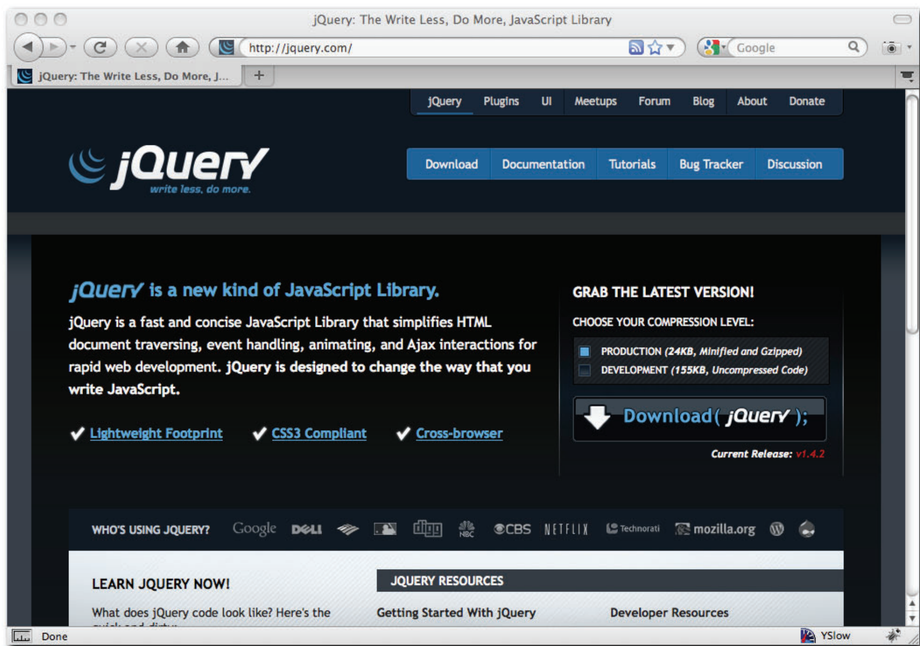


图1-1　jQuery官方网站

1. jQuery简史

jQuery发布于2006年，其创建者John Resig受够了当时那些复杂的JavaScript库，因此发明了jQuery。jQuery让Web设计师和开发者能够用更简洁的代码处理Web站点上的高级JavaScript功能。

jQuery做到了无需专业编程技能就能处理DOM，这相当了不起。当然，jQuery在某些高级领域也需要使用者具备一定的JavaScript基础，比如在表单内使用Ajax方法获取内容或提交内容

（参见第9章）、编写jQuery插件（参见第11章），以及创建移动Web站点（参见第10章）。

在过去的4年里，我认识的几乎每一位设计师或开发者都在一些领域使用或曾经用过jQuery。当我问他们为什么选择jQuery时，他们往往反问："还有比它更容易的吗？"使用jQuery是如此容易，这着实吸引了大量的用户。借助jQuery，你无需拥有计算机科学硕士学位，一样能够通过Ajax提交表单。

jQuery库擅长处理哪些事？答案是使用原生JavaScript API能做的任何事。我会在本书中深入讲解jQuery的各个功能，这里先概要地列出jQuery的主要功能。

- 事件处理：鼠标、键盘、表单及用户交互。
- 特殊效果：显示／隐藏、滑动、淡入淡出、自定义动画。
- 动画：允许使用CSS和原生效果移动页面元素。
- Ajax：使用XML或JSON与服务器端表单处理交互。
- 易于扩展：编写插件扩充jQuery核心API。
- 处理DOM。
- 处理CSS。
- 实用功能：浏览器检测及更容易使用常用的JavaScript功能。

2. jQuery用户

jQuery的主流用户群是Web设计师和开发者。jQuery广泛用于各种站点，包括中小型站点和成熟的企业站点。jQuery是自由软件，因此大多数的设计师和开发者都用它。它让那些从来没写过程序的Web设计师尝到了使用JavaScript的甜头，开始为自己的站点添加一些很酷的效果。

自从谷歌和微软开始为jQuery提供文件托管服务，jQuery风头更盛。所谓托管服务就是将文件存放到一个Web服务器上——换言之就是将文件存放到CDN（Content Delivery Network，内容分发网络）上——从而为使用该文件的站点提供更好的性能。谷歌和微软的这一举动表明，在开源的JavaScript库社区当中，jQuery当之无愧成为推荐使用的主流JavaScript库。谷歌、BBC、戴尔、美国银行、美国职业棒球大联盟、NBC和Netflix，这只是日益增多的、在开发Web站点方面使用jQuery库的一部分大公司。多年以来，Netflix这家通过直发电子邮件和网上渠道经营电影租赁业务的Web站点，一直利用高级JavaScript技术来驱动用户界面。图1-2展示了一个用jQuery创建的菜单（当鼠标悬停在电影标题上时该菜单会自动出现），这样用户无需离开这个页面就可以看到更多的信息。

3. 如何使用jQuery

jQuery很容易使用。和使用其他JavaScript库一样，需要将库包含到页面的<head></head>标签之间。下面的代码演示了如何把jQuery库包含到站点：

```
<!doctype html>
<html>
  <head>
    <script type="text/javascript" src="jquery-1.4.2.js"></script>
    <script type="text/javascript"></script>
  </head>
```

```
  <body>
    <a href="http://jquery.com/">jQuery</a>
  </body>
</html>
```

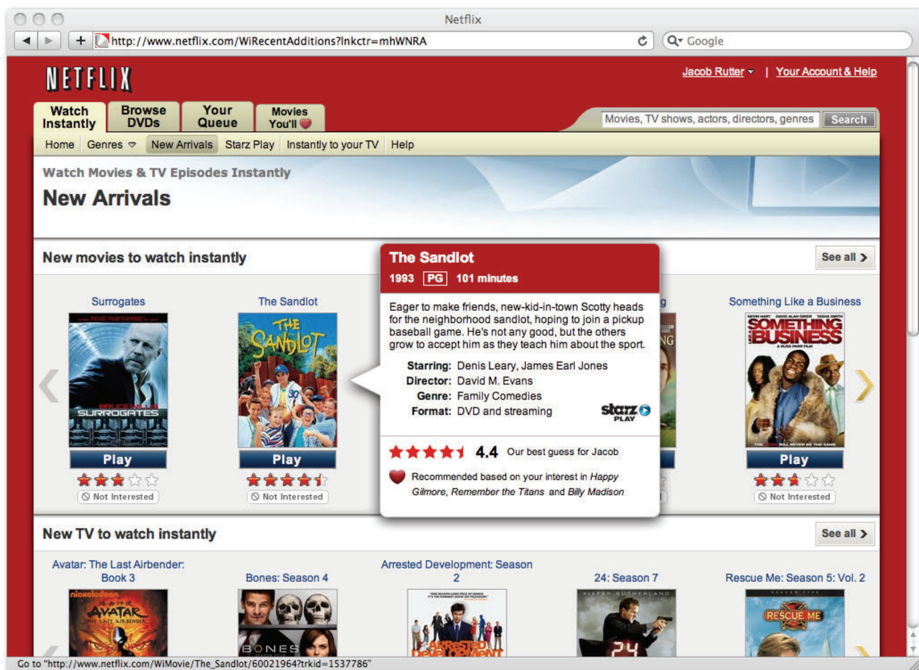


图1-2　Netflix 站点

把jQuery库添加到页面之后，你就可以使用jQuery API通过DOM访问页面的不同部分。对绝大多数Web设计师和开发者而言，这些工作他们很熟悉。如果你是一名Web设计师，很可能天天和它打交道，却没有注意到它的存在。

你还可以使用jQuery往页面中添加或从中删除HTML，并响应用户在页面上的行为，比如单击一个链接或者填写一张表单。你也可以创建动画或使用Ajax，通过Web服务将内容发送到服务器或从服务器载入内容，而根本不需要刷新页面。

4. jQuery的优势

今天的Web站点已经不再仅仅是文本、图片和到其他页面的一个个链接了。在Facebook、谷歌、微软、推特和Foursquare（我只是在这里随便说上几个）的带动下，因特网用户对Web站点的期望值越来越高。技术日新月异，而使用jQuery有助于跟上这快速的步伐。jQuery是一个有助于快速开发Web站点和应用的库，它让我们把精力集中到用户交互和界面设计上，并且无需编写冗长臃肿的代码。

因为总有一个（合适的）API可用，写起jQuery代码来远比写原生JavaScript容易。如果你熟

悉HTML和CSS编程，就很容易理解和编写jQuery代码，因为绝大多数jQuery功能都是在HTML和CSS范畴内（与用户）交互。

- 开源

JavaScript库由开源社区支持，并得到了良好的测试和及时更新。开源社区是一个巨大的支持网络。Web设计师和开发者不断地编写解决方案、写书、写插件来帮助和扩展jQuery库。

- 无与伦比的文档

目前为止，jQuery最大的优势之一是它的文档——可以说正是它造就了这个重大的库。jQuery团队在编写有关库如何工作及用户如何方便地查找API上花费了大量时间。jQuery文档站点上有一些带有完整代码示例的解决方案，还有遍布整个网络的庞大的支持团队。图1-3展示的是jQuery站点的文档子站。

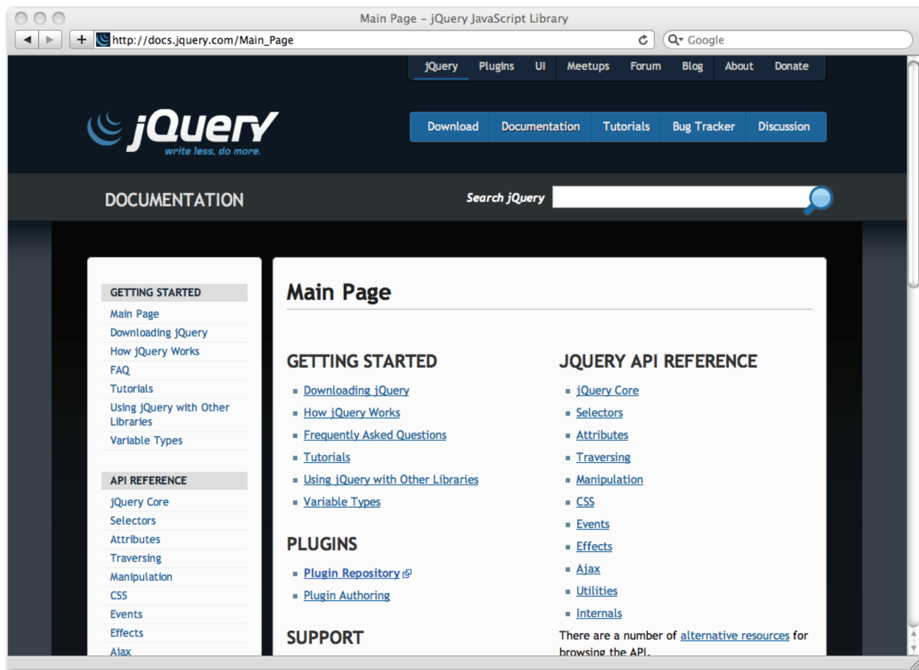


图1-3　jQuery站点的文档子站

来自社区的开发者和程序员不仅发明了jQuery，而且一直在改进jQuery，不断地发布新版本。自从2006年发布1.0版本以来，jQuery代码已经更新了23次，目前最新的版本是1.4.2。人们不停地改进jQuery，这正是它如此流行的一大原因。不经常更新的库就不那么流行。

由于库不断更新，对应的文档也就不断更新，以便提示哪些方法已经过时（不推荐使用，下一次发布时有可能被删除）并确保库可以向后兼容——也就是让文档也适用于较老的版本。每当发布一个新版本，（用户的）升级过程相当简单——把新版本的JavaScript库往服务器上一丢就行

了。当然，你应该看一眼更新日志（它概述每次发布的版本及库的改动），以查看正使用的哪些方法已经过时。

jQuery通过MIT许可证或GNU GPL v2许可证发布。基本的意思就是声明它是自由软件，只要信任jQuery插件的作者，就可以自由地使用这些代码。

- 一样的JavaScript，更少的代码

jQuery就是JavaScript：你能用JavaScript干什么，就能用jQuery干什么，一切皆有可能。我最喜欢jQuery的一点是，它提供了一个坚实的地基，而开发者又不必囿于jQuery提供的API。在使用jQuery编程时，有3种选择：

- 选用jQuery API；
- 找一个或写一个jQuery插件；
- 使用原生JavaScript。

jQuery的另一个优势是代码简洁。如果用纯JavaScript改变（一个元素的）背景色，代码会是这样：

```
document.getElementById('mydiv').style.backgroundColor = 'red';
```

使用强大的选择器引擎，jQuery达成同样目的的代码要短得多：

```
$('#mydiv').css('background-color','red');
```

这样的语法比原生语法更容易理解，设计师完全可以心里怎么想，手就怎么写。拿jQuery的语法和其他库（比如Prototype或YUI）的语法比一比，你就会明白jQuery为什么能赢得众多Web设计师及开发高手的心。jQuery的选择器引擎是它最著名，也最受人喜爱的功能。由于支持用CSS2选择器选取元素，对于那些有CSS经验的Web设计师来说上手极为容易。

- 链式调用

jQuery的另一个厉害功能是链式调用，就是说方法可以一个接一个（就像链条那样）地调用。这非常有助于保持代码短小，从而提高从Web服务器上加载并执行jQuery代码的性能。

下面是一个jQuery链式调用代码示例：

```
$('#foo').addClass('active').prev().removeClass('active');
```

下面是一个不使用链式调用的代码示例：

```
$('#foo').addClass('active');
$('#foo').next().removeClass('.active');
```

看看这两个例子，使用链式调用编写的jQuery代码更干净、更简洁。本书中的代码示例大都使用了链式调用。

- 兼容各种浏览器

随着Safari、Firefox、IE、Google Chrome和Opera的不断更新，让页面支持所有的主流浏览器便成了重中之重。浏览器战争已经成为每一位Web设计师天天苦恼的问题。

如果使用的是jQuery，你尽可放心，它兼容所有主流浏览器，如IE 6.0+、Mozilla Firefox 2+、Safari 3.0+、Opera 9.0+和Google Chrome。

使用JavaScript经常会遇到的一个大问题是，你不得不编写不同的代码以支持不同浏览器。一些Web设计师和开发者选择为特定浏览器编写专用样式表，来支持不同浏览器，一般是专门为IE写一个，为其他浏览器写一个。JavaScript也有类似问题。下面的代码展示了（使用原生JavaScript）如何在不同的浏览器中生成Ajax请求对象：

```
if(window.XMLHttpRequest) {
  xhr = new XMLHttpRequest();//面向Firefox/Safari的代码
} else if(window.ActiveXObject) { //Active X版本
  xhr = new ActiveXObject("Microsft.XMLHTTP"); // For IE
}
```

使用原生JavaScript，同样的功能你不得不写两个不同的函数，测试它们，为它们"捉虫"（修复bug）。这已经让代码难于管理，更不用说你还会遇到这样的情况了：为了让一个功能支持多个浏览器，不得不分别写好几个函数。有一个方法可以彻底摆脱这个痛苦，那就是使用jQuery：同样的功能只需要写一次，就能支持所有的浏览器。

作为对比，下面这行代码展示了在jQuery中发起一个Ajax请求有多么容易。

```
$.ajax();
```

- 兼容CSS3

所有的现代浏览器都支持CSS1和CSS2（CSS的前两个版本），并且绝大多数Web设计师和开发者现在都在用CSS2。CSS3正在开发当中，它提供了一些增强特性，如内嵌字体、圆角、高级背景图片功能、颜色、文本特效、渐变等。现在只有少数几个浏览器能完全支持2010年7月发布的CSS3标准，它们是Firefox 4、IE 9、Opera 9和Safari 4。这些浏览器的一些老版本对CSS3提供部分支持。

jQuery目前只支持CSS3标准中的新选择器。这意味着什么？CSS3的新功能之一——新添加的属性选择器（作为对CSS2属性选择器的补充和增强），与jQuery中已有的属性选择器非常接近。这些选择器让你能够基于属性对内容添加样式，这样就能过滤出属性中特定的值。请看下面的代码：

```
p[title=*foo] {background:black;color:white}
<p class="text" title="food is good foo you">This is my sample text</p>
```

- 不突兀的JavaScript

如下例所示，很多人在a标签的`href`属性中直接嵌入JavaScript代码以创建弹出窗口。这个代码最大的问题在于把代码直接放到了链接`href`属性当中。如果一个用户碰巧禁用了JavaScript（虽然这确实有点罕见），这个链接就失效了，并且也没有退路让这个用户看到帮助信息。

```
<a href="javascript:window.open('help.html', 'help window',
'height=800,width=600,toolbar=no');return false;">Help</a>
```

这是突兀地使用JavaScript的一个例子。对Web设计师来说，这就像在编写内联样式，而不是将样式从内容中分离到表现层。为了和这个突兀使用JavaScript的例子对比，下面是一个使用JavaScript的、不突兀的例子，它使用了jQuery，代码也与上面的例子相似。如果JavaScript功能被禁用，这个版本虽然一样无法执行封装在`click`函数中的代码，但提供了让用户单击链接以访问帮助页面的候补方案。

```
<!doctype html>
<html>
  <head>
    <title>Unobtrusive jQuery</title>
    <script type="text/javascript">
    $(document).ready(function (){
      $(".help-link").click(function() {
        var linkHref = $(this).attr('href');
        window.open(linkHref,'help window', 'height=800,width=600,toolbar=no');
        return false;
      });
    });
    </script>
  </head>
  <body>
    <a href="help.html" class="help-link">Help</a>
  </body>
</html>
```

优雅降级和渐进增强是两个用来支持浏览器新特性及旧浏览器的两个（常用）策略，可用于提供最佳的用户体验。渐进增强策略比较新，不过二者的区别主要在于采取的方式不同。接下来我将详细介绍这两个策略。

- 优雅降级

使用优雅降级方案，Web站点在所有新式流行浏览器中都能正常工作，如果用户使用的是老式浏览器，则代码会检查以确认它们是否能够正常工作。由于IE独特的盒模型布局问题，绝大多数Web设计师和开发者都通过专门的样式表或针对不同版本的IE的hack实践过优雅降级了（哦，请别指责IE6）。

```
<a href="javascript:window.open('help.html', 'help window',
'height=800,width=600,toolbar=no');">Help</a>
<noscript>
Please upgrade your browser or turn on JavaScript, as your browser is not working with
our Website.
</noscript>
```

- 渐进增强

渐进增强指的是这样一种策略：从被所有浏览器支持的基本功能开始，逐步地添加那些只有新式浏览器才支持的功能。渐进增强是值得所有开发者采纳的做法，能让Web站点的可用性更好。如果你总是交付一个所有人都能用的基本功能集，加上只支持先进浏览器的专用升级版功能集，比如那些采用了CSS3和HTML5技术的功能集，用户满意度会更高。目前仅Safari 4和Opera 10.6支持HTML5和CSS3。

渐进增强方案并不假定所有有户都支持JavaScript，而总是提供一种候补方法，确保用户可以访问（主要的）内容。考虑一下“不突兀的JavaScript”那节中的弹出窗口，如果用户不支持JavaScript，我们就通过a标签中的`target`属性告诉浏览器，让它在新窗口中打开这个链接（而不再使用弹出窗口）。几乎所有的浏览器都支持这个。

```
<a href="help.html" target="_blank">Help</a>
```

在本书中，我致力于使用渐进增强策略让使用现代浏览器的人享有较好的用户体验，同时为那些使用老式浏览器的人保证基本的用户体验。

- 不突兀的JavaScript与jQuery

因为所有的jQuery（JavaScript）代码都独立于HTML（存放到一个外部JavaScript文件中）或集中存放在HTML文件的头部区域，除非你使用托管在CDN上的jQuery文件（托管解决方案），所以jQuery使得实践这些策略（优雅降级和渐进增强）更加容易。由于HTML元素中不嵌入任何JavaScript代码，只要开发者牢记这些实践，在架设站点时预留退路（非JavaScript方案）总是可行的。

第 2 章 jQuery入门

由于（各种库中）jQuery入门最容易，它成了众多设计师和开发者的选择。开发者需要下载一份JavaScript库文件的副本，并使用一个`script`标签将它包含到网页的头部。所有的JavaScript库都一样，无一例外：必须在网站或应用中先包含库，然后才可以使用库的特性。

本章会指导你搭建一个本地开发环境（可选的），选择并下载合适的jQuery库文件，然后将它包含到站点当中。我还会解释jQuery包装器的用途。

2.1 搭建开发环境

搭建开发环境的第一步是选择喜欢的编辑器。有很多流行的编辑器，如Dreamweaver、Coda、TextMate，还有EditPlus等。我主要使用Coda，这是一个Mac OS X应用程序，是供设计师和开发者开发Web站点和应用的专用工具。这个软件集成了很多实用的功能，包括FTP、终端（命令行）、文件管理、CSS及代码编辑器、语法高亮、自动完成、增强的查找和替换功能、预览及多语言支持。如果不喜欢使用代码编辑器，你也可以用Windows上的记事本或者苹果系统上的TextEdit，只是这样就不能使用代码编辑器所提供的丰富特性了。

在开始编写jQuery代码之前，我们需要有个地方测试代码，也就是说，我们需要一个开发环境。开发环境可以是一个本地环境（包括一个本地的Web服务器和Web浏览器），也可以是一个远程Web主机。我们主要使用它在一个仿真的环境中测试代码。使用Coda或Dreamweaver等程序的好处是，这些软件本身支持设置外部Web主机，这样我们就能直接在外部服务器上工作，轻松地进行测试。

有人会说只需要一个本地文件夹就能干活，在浏览器中逐个打开文件进行测试。这样做不是不行，但不可能提供真实环境那样精确的预览。以这种方式开发完某些jQuery功能，也许等到上线你却发现得到的是不同的结果。我坚信自开发伊始就拥有一个仿真度极高的环境，再好不过了。

本地开发环境很容易搭建，它在你不能上网时尤其有用。对Mac用户来说，搭建开发环境最流行的选择是MAMP（www.mamp.info/），这4个字母代表Mac/Apache/MySQL/PHP（参见图2-1）。这是一个一揽子解决方案，安装并运行它可以在本地进行测试，就像你在操作一台真正的Web服务器一样。Windows用户可以使用Wamp Server，也就是Windows/Apache/MySQL/PHP Server

（www.wampserver.com/en/）。我建议尽量将Apache用作Web服务器，它是一个非常稳定的开源Web服务器，主要运行在Linux平台上。

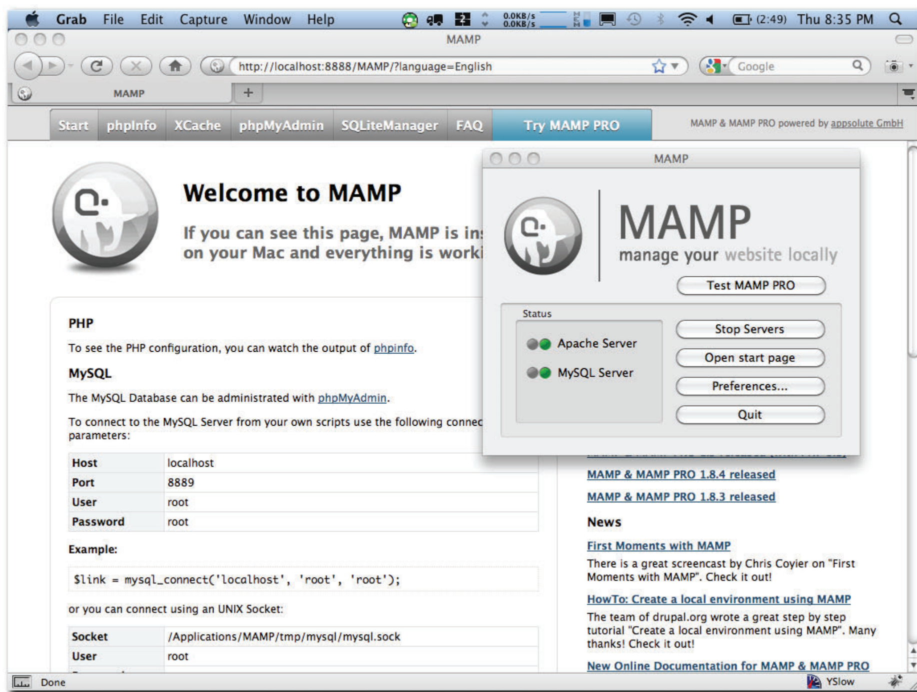


图2-1　我的桌面上正在运行的MAMP（另见彩插图2-1）

对Mac OS X用户来说，也可以使用系统内建的Apache Web服务器。要在Mac系统上设置Apache服务器，需要以下步骤：

(1) 打开System Preferences（系统设置），你会看到一些小图标，单击这些图标可以设置Personal（个人信息）、Hardware（硬件）、Internet and Wireless（因特网和无线网）和Other（其他项）。

(2) 单击Sharing（共享）图标打开Sharing（共享）面板，然后在共享服务列表中选中Web Sharing（Web共享）选择框。Sharing面板显示的这些选项可以用来设置文件、屏幕、打印机共享等。如果Web Sharing选择框已经选中，就跳过这一步。

(3) 如图2-2所示，在确保Web Sharing选择框选中之后，我们现在启动Apache Web服务器。你会看到面板的右侧有一个红色状态图标变成了绿色，并看到这样的文本：Your personal website，in the Sites folder in your home folder，is available at this address：http：//xx.xx.xx.xx/~yourname（你的个人站点位于你Home目录的Sites子目录中，并可以通过http://xx.xx.xx.xx/~yourname访问）。

(4) 单击IP地址会启动默认Web浏览器，并打开默认页。站点的文件（HTML、CSS、JavaScript、Images）都在你的～/Sites目录中。如此这般，一个Web服务器就设置好了。我所有的本地开发工作都是用这种方式完成的。

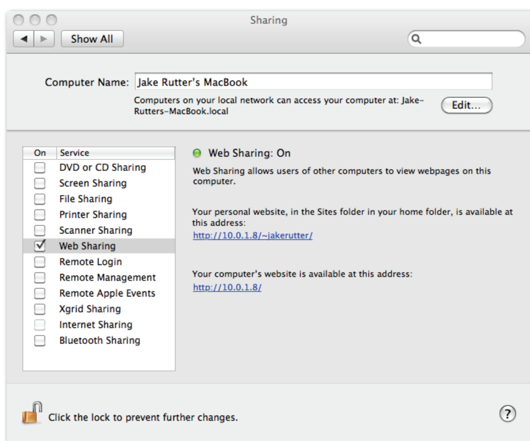


图2-2　Mac OS X中的共享设置对话框（另见彩插图2-2）

在Firefox中使用Firebug扩展

如果尚未将Firefox用作主要的开发用浏览器，那么在进一步学习本书内容之前，我强烈建议你下载并安装一份。本书中所有例子用的都是Firefox 3.6.8和Firebug 1.5.4。2006年12月，Joe Hewitt发布了他编写的Firefox扩展Firebug[①]，如今它已是一个开源项目。从那时起Firebug不断升级，推陈出新，现在已经有100多万开发者在使用Firebug。

Firebug提供了许多小工具，是广大Web设计师和开发者处理HTML、CSS和JavaScript的利器。Firefox不但免费而且开源，任何人都能通过Firefox扩展网站（addons.mozilla.org）下载并使用它。

Firebug不但查看代码方便，还支持直接在页面上修改HTML和CSS代码并立即查看修改结果（参见图2-3）。它还集成了一个非常强大的JavaScript调试器，能有效地帮助我们找出bug。Firebug控制台有一个非常棒的功能，我们可以直接在控制台内执行JavaScript脚本（当前页面环境下），这实在方便极了。

1. 安装与启用Firebug

执行以下步骤，安装并启用Firebug。

(1) 打开Firefox浏览器，访问www.getFirebug.com。

(2) 单击Install Firebug for Firefox（为Firefox安装Firebug）按钮。

(3) 接着会出现一个提示窗口，提示你只可以安装值得信任的扩展（参见图2-4），同时你会看到Install（安装）按钮上有一个倒计时的数字，当这个数字变成0时，Install Now（现在开始安装）按钮就可以单击了。此时请单击它。

① 此扩展的版本0.2发布于2006年1月12日，版本1.0发布于2007年1月24日。（本书脚注如无特殊说明，均为译者注。）

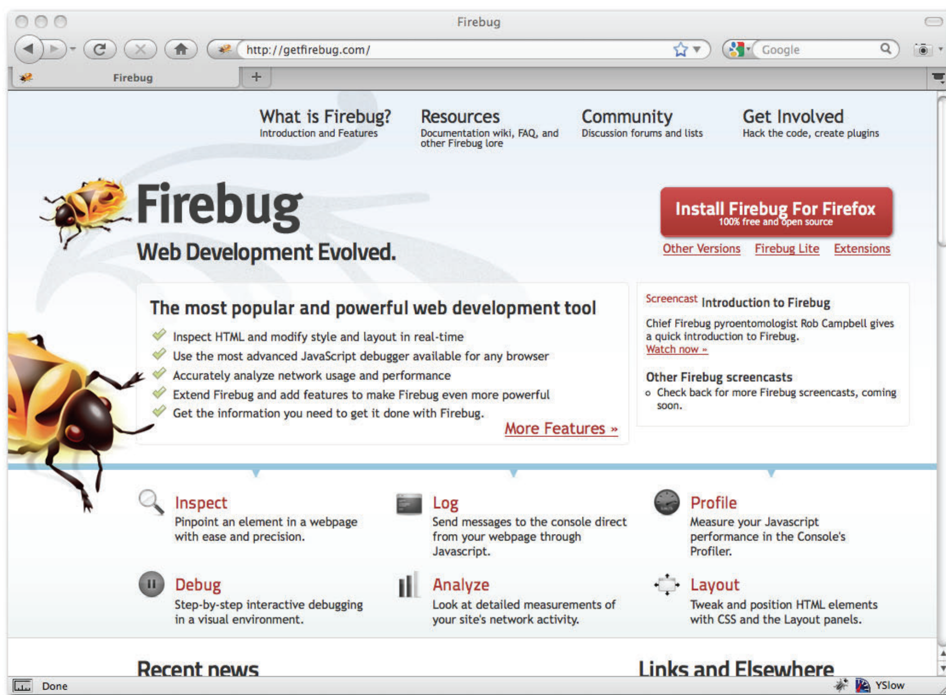


图2-3　Firebug官方网站

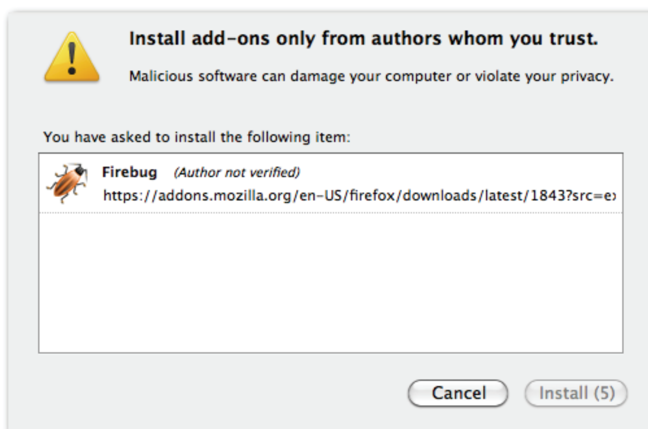


图2-4　Firebug安装提示

(4) 接着你会看见一个安装进度条，表示插件正在安装到浏览器上。等插件安装完毕，你会看到一条确认消息和一个Restart Firefox（重启Firefox）按钮（参见图2-5）。

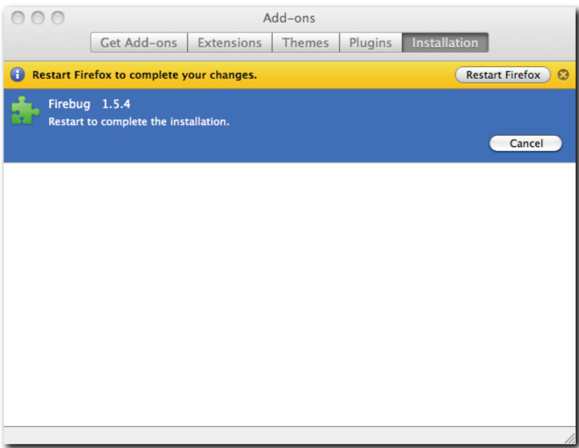


图2-5　Firebug安装结束窗口

(5) 祝贺你，Firebug已经安装好，可以使用了！图2-6展示的是安装过程的最后一步。

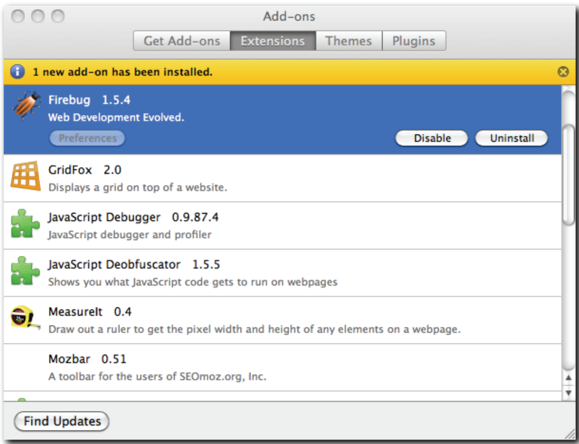


图2-6　Firebug安装完成并重启浏览器之后的Firebug安装确认提示

2. 启用Firebug

执行以下步骤，启用Firebug。

(1) 在Firefox中打开一个页面。出于演示目的，我打开了www.mozilla.com页面。

(2) 等页面加载完毕之后，有好几种方法打开Firebug。最容易的方式是单击位于浏览器右下角[①]的Firebug图标。图2-7演示了一个安装好了的Firebug例子，你会看到浏览器右下角有一个Firebug图标。

① 最新浏览器和最新Firebug很可能改变了这个位置，如果你在浏览器右下角看不到Firebug图标（一个小萤火虫），别犹豫，按F12键一样能激活Firebug。

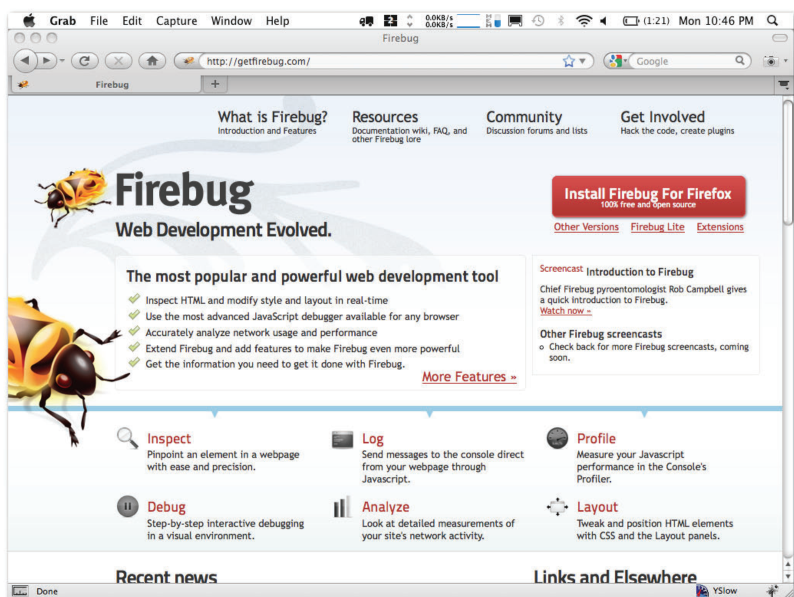


图2-7　一个网页（Firebug已安装好）

你也可以在浏览器窗口中单击鼠标右键，然后在弹出菜单中选中Inspect Element（检视元素）。图2-8演示了一个浏览器窗口中弹出的菜单。

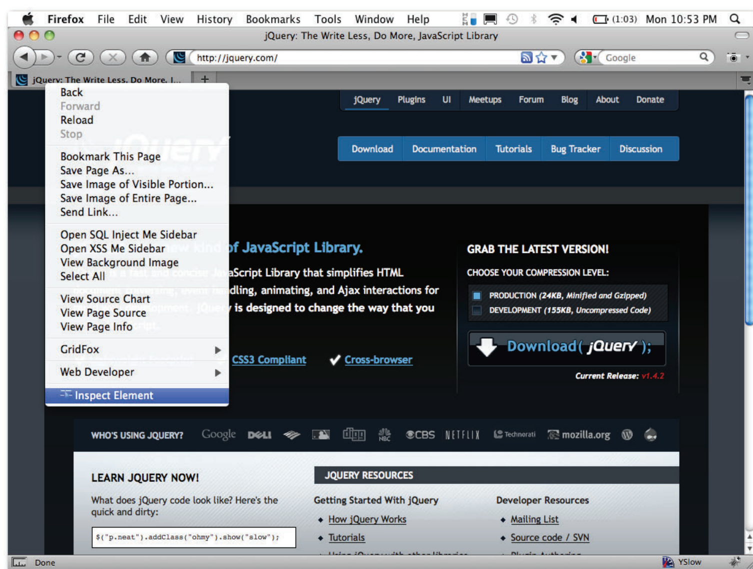


图2-8　通过Inspect Element打开Firebug

(3) 打开Firebug之后，你会看到一排标签：Console（控制台）、HTML、CSS、Script，还有DOM等。

3. 检视与编辑HTML

Firebug的检视和编辑功能实在太强大了，尤其是DOM的即改即现功能，这让HTML和JavaScript调试工作变得简单很多。如果你的脚本添加或修改了一段HTML，打开检视窗口，就能清楚地看到HTML改变的“直播过程”。当我需要调试JavaScript时，第一步总是打开检视窗口，先确认即将操作的HTML已创建好。图2-9演示的是一个打开了的Firebug窗口，其中选中了body标签。

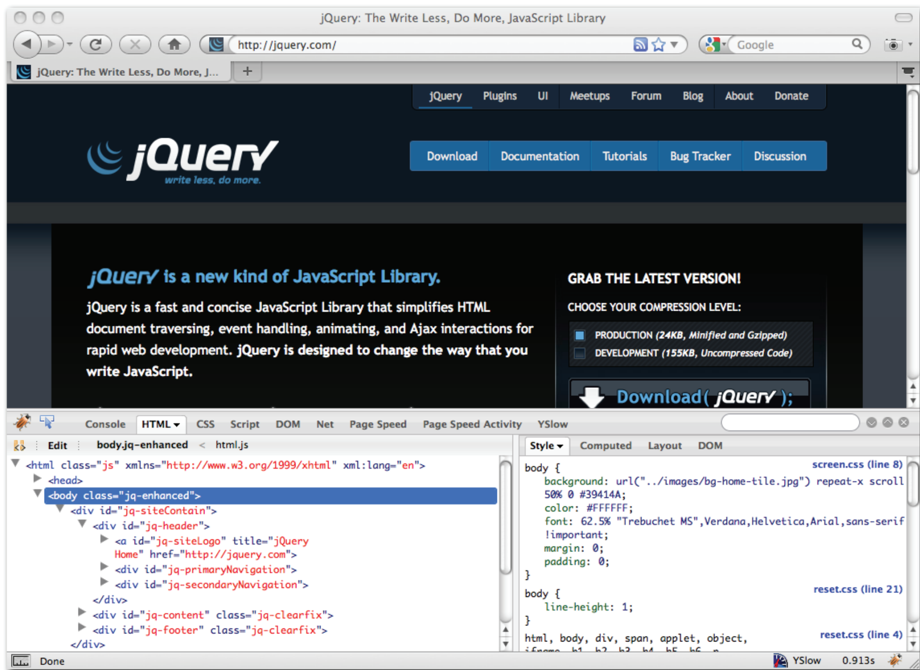


图2-9　Firebug的HTML编辑区域

4. 使用控制台

调试JavaScript的第二步是使用Firebug控制台。我在修复了DOM问题之后，接着就用控制台来测试我的脚本（在这个页面上）。你会看到控制台分两个区域，其中左边的区域用来显示错误信息，而右边的区域用来输入JavaScript指令。①

① 如果Firebug刚刚安装好，单击Console标签，你看到的应该是上下两个区域。上面比较大的区域显示控制台信息，下面很窄的区域（只有一行）用来输入JavaScript语句，这个区域一次只能输入一行代码，且其右侧有一个圆形按钮，按钮上有一个上箭头。这个按钮用来激活多行测试区，单击这个按钮，上下两个区域就变成作者所说的左右两个区域了。

(1) 打开Firebug，单击Console标签。

(2) 如果页面上的脚本有错，它们会显示在左边的区域。图2-10展示了Console的两个区域，以及显示在左边区域中的错误信息。

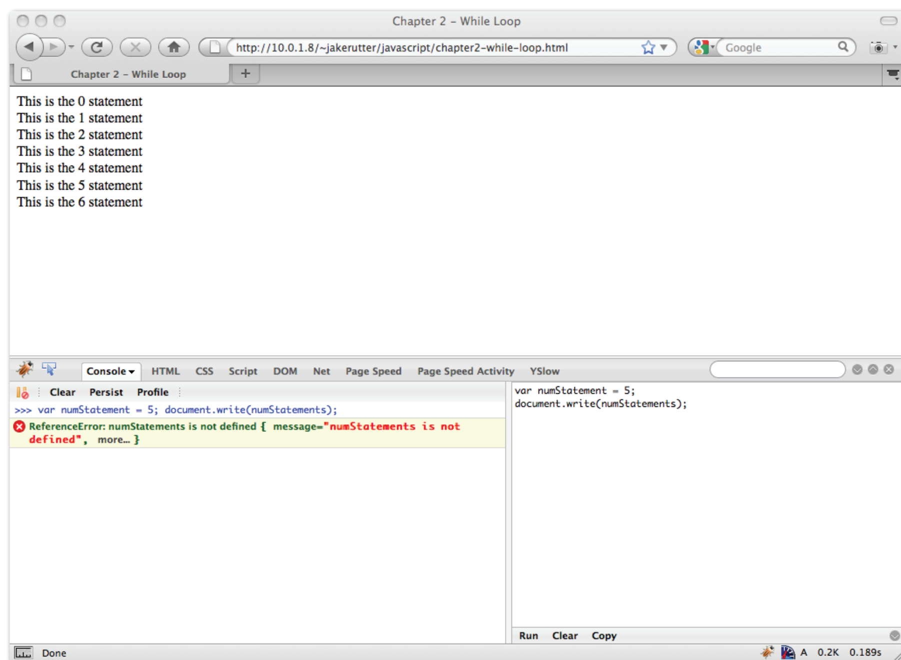


图2-10　Firebug控制台及其中显示的错误信息

5. 在Firebug控制台中测试JavaScript脚本

使用Firebug控制台，我们无须将脚本加载到页面，就能直接在Web服务器上进行测试。这实在是一种了不起的JavaScript调试方法。如果JavaScript代码运行出错，Firebug控制台会及时给出错误信息。在把脚本写入HTML文件之前能以这种方式对脚本进行充分测试，这实在太棒了。

6. 使用Firebug进行高级JavaScript调试

要调试那些更复杂的JavaScript应用程序，Firebug有专门的JavaScript调试器。这个调试器功能强大，支持在脚本的不同位置添加断点，这样就可以通过暂停、停止、继续执行脚本仔细观察变量和对象。这个调试器是专为高级JavaScript程序员准备的，因此在本书中我不会详细介绍它。

7. 在其他浏览器中调试JavaScript

地球上并非只有Firefox浏览器提供了Web开发工具。苹果公司的Safari、谷歌公司的Chrome以及（新版的）IE都提供了类似的工具，但是它们中任何一个都无法与Firefox的Firebug插件相比。Safari/Chrome调试器提供了一些与Firebug类似的功能，包括元素检视功能和资源管理标签，但是它们缺少一个像Firebug包含的那样强大的JavaScript调试器。其他Web开发工具我用得不多。我偶尔使用IE/Safari（开发工具）检查页面，而Firebug一直是我Web开发工作最主要的帮手。

在学习jQuery和其他JavaScript库之前，作为Web设计师或开发者的我们需要熟悉JavaScript的基本概念。也就是说，没有坚实的JavaScript基础，我们只能使用和写一些简单的脚本，而深入了解javaScript工作原理却能够大大提高开发效率，增加对JavaScript这门语言的理解，有效提高生产率。

2.2　下载 jQuery 库

在开始jQuery开发之前，我们首先要去jQuery站点下载jQuery库。jQuery库就是一个JavaScript文件，可以用两种方法访问：

(1) 下载jQuery.js并将它部署到你的Web站点；

(2) 使用存放在CDN（内容分发网络）上的某个托管版本。

我建议你下载一份jQuery副本，并将它部署到自己的计算机上，用于开发和测试。在没有因特网连接的时候，这么做尤其有用。执行以下步骤下载jQuery：

(1) 在浏览器中打开jQuery官方网站www.jquery.com。

(2) 单击页面顶部主导航中的Download（下载）链接，它会把你带到一个页面，这个页面提供了许多种访问jQuery库的方式。图2-11即jQuery下载页。

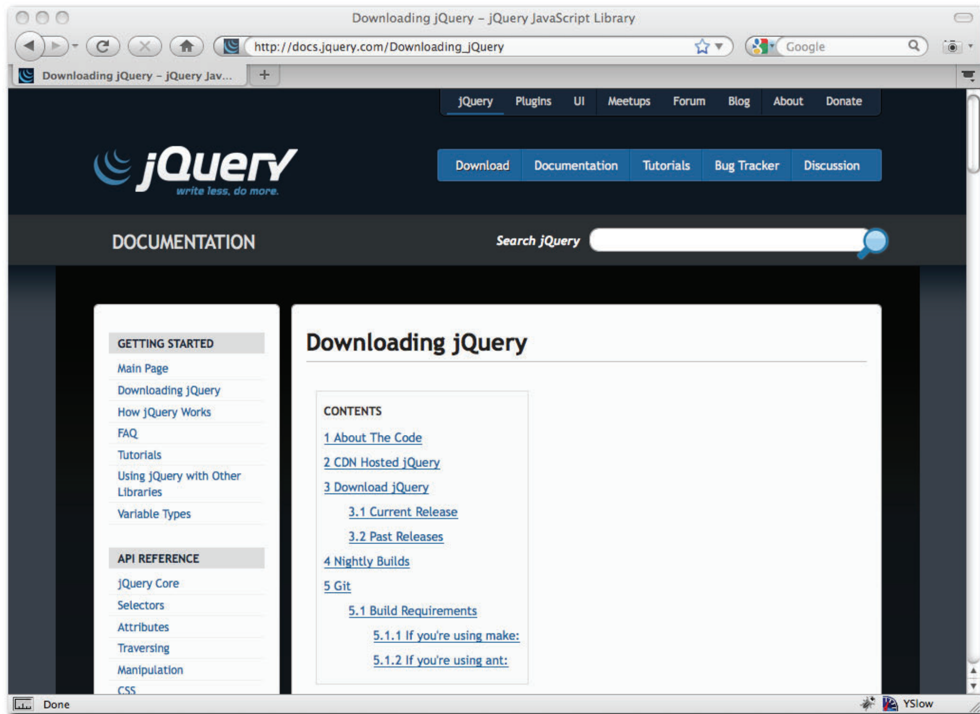


图2-11　jQuery官方网站下载页

很多网站使用免费的CDN服务，因为这些CDN服务已经被证明既可靠又快速，能节省站点消耗的带宽。CDN（Content Delivery Network，内容分发网络），是由一些网络公司巨头（如谷歌、微软、Akamai等）提供的内容分发方案。

CDN网络不但速度快，而且几乎没有流量限制，它们在很多点上都部署有jQuery库。当一个用户通过浏览器访问你的站点时，CDN能根据该用户的IP地址选择离他最近的服务器分发jQuery库给他，这会有效降低页面载入时间。图2-12展示的是托管在谷歌CDN上的Ajax库列表页，这个页面介绍了很多包含jQuery或其他库到站点（或应用程序）的选项。

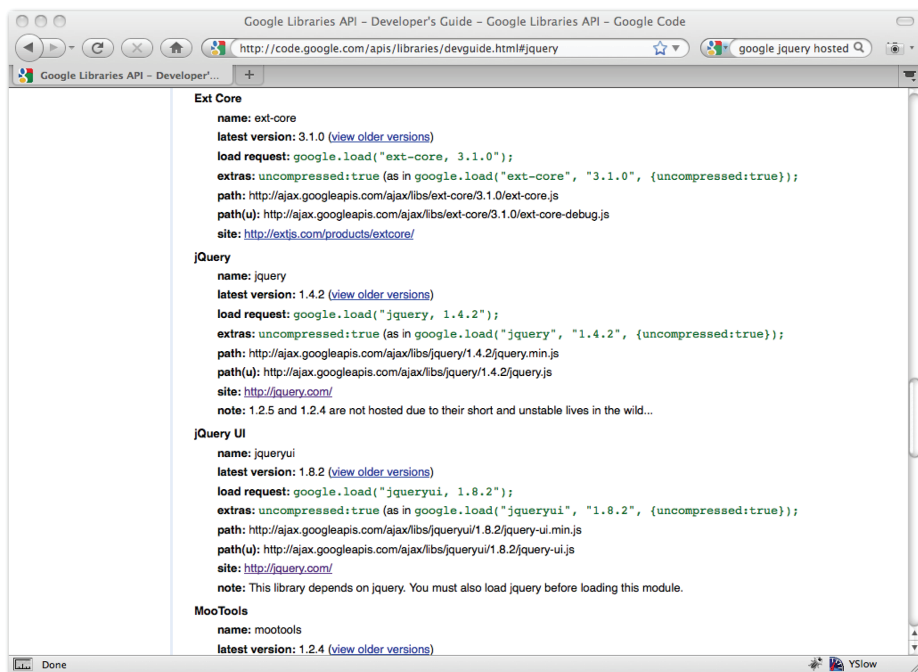


图2-12　托管在谷歌CDN上的Ajax库列表页

如果使用托管方案，你可以在URL中指定要使用的jQuery版本。要使用托管在谷歌上的托管jQuery，可参考下面这行代码：

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js">
 </script>
```

谷歌和微软都是有着巨大市场份额的大公司，已经有许多站点在使用托管在它们服务器上的jQuery库。使用托管库的站点越多，你在初次访问一个站点时，该站点用到的托管库文件已经缓存在你的浏览器中的可能性就越大。（当你访问一个网站时，浏览器会缓存从服务器上获取的文件。）

举例来说，我每天都用Digg（www.digg.com），Digg是一个新闻门户站点，它根据用户投票

多寡显示新闻。这个站点使用的是托管在谷歌服务器上的jQuery库，因此我决定把jQuery库加到我自己的站点上。在我访问自己的站点时，请求jQuery库时浏览器会从浏览器缓存中获取一个已缓存的版本，除非从谷歌那里得到了一个库已经被修改过的响应。如果是那样，浏览器就放弃缓存中的版本，而去谷歌服务器获取最新版本。

表2-1是两种jQuery库对比表。

表2-1　jQuery库版本

格　式	描　述
未压缩版本（大约155 KB*）	主要用于开发过程，方便调试，也方便高级程序员随时看看jQuery的某个功能是怎么实现的
压缩版本（大约55 KB*）	体积非常小，用于生产环境。采用压缩技术去除了所有不必要的字符，如注释、换行符、不必要的空格和制表符以缩短载入时间

* 这里列出的文件大小对应的是1.4.2版本的jQuery。

一些老练的用户可能希望通过Git（一种版本控制软件）获取最新版本的jQuery库。在这本书里我不会详细介绍Git，如果你想深入了解Git，不妨访问www.git.com。

2.3　在页面中包含 jQuery 库

做出决定以后（或是下载一份jQuery库，或是使用托管版本的某个jQuery库），我们都需要在页面中包含它。我们可以使用`<script>`标签将jQuery库包含到HTML页头的`<head></head>`标签之间，也可以将jQuery库包含到刚好在`</body>`标签之前。

我们必须在包含jQuery库和自己编写的jQuery代码之前先包含CSS（层叠样式表），只有这样才能确保在jQuery修改DOM之前所有CSS已经应用到DOM（Document Object Model，文档对象模型）。

在网页的HTML的开头应该包含一份正确的`doctype`定义。没有`doctype`的页面在浏览器中可能会有各种不确定的怪异行为，也会使页面无法通过代码验证检查。合适的`doctype`有助于代码在不同的浏览器中表现一致。在缺少`doctype`的页面中，jQuery有可能出现渲染错误，CSS也一样会渲染出错。

本书中所有的例子都使用HTML5 `doctype`以确保在较老的浏览器中实现渐进增强和优雅降级（如果你忘了什么是渐进增强和优雅降级，请阅读第1章）。与早期那些很难记忆的`doctype`相比，HTML5 `doctype`设置起来非常简单。

下面的代码就是HTML5 `doctype`：

```
<!DOCTYPE html>
```

下面是一个非常简单的HTML文档，这个文档告诉我们如何把jQuery包含到页面的头部。记着总是先包含所有CSS文件，这样才能确保在修改DOM之前页面已经被正确渲染。

```
<!DOCTYPE html>
<html>
  <head>
    <title>My jQuery Example</title>
    <link href="css/global.css"/>
    <script src="js/jquery.js" type="text/javascript"></script>
    <script type="text/javascript">
    //脚本
    </script>
  </head>
  <body>
  </body>
</html>
```

另一种方式是把jQuery库包含到页尾。把库包含到页尾会在某种程度上提高页面加载速度，这是因为JavaScript不会阻塞页面其他部分的加载（在页头加载脚本会阻塞页面的渲染）。另外，这样也能确保在JavaScript起作用之前DOM已经加载完。

```
<html>
  <head>
    <title>My jQuery Example</title>
    <link href="css/global.css"/>
  </head>
  <body>
  <h1>Hello jQuery!</h1>
  <div id="page-container">
    <p>You can place your jQuery at the end of the page too!</p>
  </div>
  <script src="js/jquery.js" type="text/javascript"></script>
  <script type="text/javascript">
  //脚本
  </script>
  </body>
</html>
```

如果要在页面上包含托管在谷歌 CDN上的jQuery，就一定不能使用相对路径，而是要使用谷歌提供的绝对路径：

```
<html>
  <head>
  <title>My jQuery Example</title>
  <link href="css/global.css"/>
  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"
    type="text/javascript"></script>
  <script type="text/javascript">
  //脚本
  </script>
  </head>
  <body>
  </body>
</html>
```

2.4 理解 jQuery 包装器

在开始jQuery编程之前，我们有必要搞清楚jQuery包装器是怎么回事，以及它如何作用于DOM。在大多数编程语言里，包装器包装某个东西（通常是一个对象）以扩展其功能。本质上jQuery包装器使用选择器将自身附加到DOM上，从而达到扩展DOM的目的。实际上jQuery并没有提供什么新方法，它只是使用原生JavaScript方法作为“米”，做出了极为美味（极其易用）的一顿“大餐”。

与原生JavaScript相比，jQuery包装器的威力在于只需要极少的代码就能实现同样的功能。下面是一个jQuery选择器语句示例：

```
$(selector)
```

jQuery提供了许多事件方法，`document.ready()`事件处理方法是其中最重要的一个，它只在DOM完全加载之后触发。（在JavaScript语言中）方法仅仅是函数的另一种表达方式。而在其他面向对象编程语言中，方法比函数更强大。jQuery最强大之处在于操作DOM，因此在对DOM做任何操作之前，我们必须确保DOM已经准备好。

应该把所有jQuery代码都放到`document.ready()`事件处理函数当中，这样就能保证它们在DOM准备好之后才开始执行。这个事件类似于JavaScript原生的`onLoad`事件，不过`document.ready()`仅在DOM完全准备好之后触发。

以下示例代码以HTML内嵌脚本的方式，演示了如何使用`document.ready()`方法：

```
<html>
  <head>
  <title>My jQuery Example</title>
  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"
  type="text/javascript"></script>
  <script type="text/javascript"></>
  $(document).ready(function() {
  //脚本
  });
  </script>
  </head>
  <body>
  </body>
</html>
```

每个开发者都应该养成使用`document.ready()`方法的习惯，并把自己编写的jQuery代码存放到单独的脚本文件当中。这是我推荐的工作方式，有助于将脚本代码与HTML代码分离。我习惯把自己的jQuery代码保存到jquery.function.js当中，并把它最后一个包含到页面当中（在所有jQuery核心库文件之后）。

`document.ready()`有一种简写形式，如果你想少打几个字[1]，或者提高移动应用符的性能，

① 省这么几个字符显然是无法显著提高性能的，不过能帮程序员省点打字时间是真的。作者在这里这么讲是和大家开个玩笑。

不妨使用这种形式。下面是使用简写版本的例子：

```
<html>
  <head>
  <title>My jQuery Example</title>
  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"
  type="text/javascript"></script>
  <script type="text/javascript"></>
  $(function() {
  //脚本
  });
  </script>
  </head>
  <body>
  </body>
</html>
```

要想解释清楚jQuery包装器，我需要先解释清楚document.ready()语句都会干些什么。首先说说美元符号（$），大家知道它是jQuery的别名，紧随其后的是选择器参数。选择器需要用一对括号括起来。在下面的例子中，我传入document对象作为DOM选择器。jQuery别名（$）及其选择器参数构成了jQuery包装器。

```
$(document)
```

在选择器表达式之后不光可以绑定ready事件，也可以绑定其他事件。

```
.ready()
```

这里的function函数体用来存放我们的代码，这些代码将在DOM加载并准备好（这时图片尚未加载完毕）后执行。之所以把这个函数放到ready方法的一对小括号之间，是因为ready事件（不仅仅是ready，所有的事件绑定都）要求其参数为一个函数：

```
.ready(function() {
  // jQuery DOM 代码
  alert("The DOM is fully loaded and ready");
});
```

window.load方法与document.ready非常相似，但它要等待所有的图片加载完成之后才会执行用户的jQuery代码：

```
$(window).load (function){
    //jQuery 代码
    alert("The window has been loaded");
});
```

2.4.1　在document.ready事件处理方法之外执行代码

绝大多数用户jQuery代码都应该放到document.ready()事件中执行。但是有些原生的JavaScript代码（如变量、数组的赋值等类似语句）例外，因为它们并不需要等到DOM准备好以后才执行，与DOM也没有任何交互。

下面例子中的代码必须等到DOM准备好之后才能执行,因为其添加新内容的行为依赖DOM。这段脚本共初始化了3个变量，其中前两个在document.ready()事件处理方法之外，第三个在document.ready()之内，因为添加内容的for循环要用到它。

```
<!doctype html>
<html>
  <head>
    <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.
      min.js"></script>
    <script>
    var numShows = 10;
    var numTickets = 100;
    $(document).ready(function() {
     for(i=0; i < numTickets; i++) {
       var numTotal = i + 1;
     $('.container').append("<p>There are " + numTotal +
                              " tickets available</p>");
     }
    });
</script>
<body>
  <div class="container">
  </div>
</body>
</html>
```

2.4.2　防止与其他库发生冲突

在编写使用jQuery的代码时，如果没有适当的预防措施，很可能会与其他JavaScript库发生冲突。绝大多数冲突都与$别名有关，因为Prototype库也使用$别名。要避免与别的库发生冲突，我们需要采取以下两个步骤。

(1) 在jQuery库代码的最后添加一行noConflict方法调用。为了让别的库能正常工作，noConflict方法使jQuery的代码不再依赖这个别名。

```
$.noConflict();
```

(2) 修改所有用到jQuery的代码，将其中的$别名改为jQuery，如下例所示，将：

```
$(document).ready() {
   //代码
});
```

改成这样：

```
jQuery(document).ready() {
   //代码
});
```

如果不想使用别名jQuery，你也可以自定义一个别名，这只需要一行JavaScript代码。在下面的例子里，我不再使用$，而是定义了新的别名$alien。就这么简单！

```
var $alien = jQuery;

$alien(document).ready() {
   //代码
});
```

2.4.3　用jQuery写JavaScript

变量是储存信息的有效方式，特别在写JavaScript时。我在写jQuery代码时经常使用变量，在本书后面的例子里，我也会经常这样做。在jQuery中使用变量和在JavaScript中使用变量并无不同。给变量赋值，然后在jQuery包装器中引用它们，这本来就是最基础的JavaScript内容。

jQuery之美在于它就是JavaScript，如果你有一些JavaScript知识，不用迟疑，这些知识完全都可以用到jQuery上。你不必担心即将学习的新语法、新代码习惯和新方法，因为除了新语法更容易理解和学习之外，jQuery的绝大多数功能都基于原生JavaScript。

你可能会困惑，如果jQuery就是JavaScript，为什么不直接学习JavaScrpit？答案是：jQuery能做JavaScript能做的一切，而且做起来容易百倍。jQuery的口号是“写得更少，做得更多”——一点儿也没错。不需要特别了解JavaScript，你就可以将20行原生JavaScript代码转换为5行jQuery代码。如果对JavaScript有着足够的好奇心，学习jQuery将有助于你了解JavaScript API。

JavaScript难以理解。有了jQuery，Web设计师不必了解复杂的JavaScript就能用JavaScript API实现想要的功能。jQuery确实打开了一扇门，让许许多多缺少编程经验的Web设计师也能为Web站点添加一些交互。这是一个使用jQuery的最好的时代：jQuery社区对jQuery的支持正以惊人的速度成长。

Part 2

第二部分

jQuery 基础

本部分内容

第 3 章

jQuery核心：选择器、过滤器及CSS

（灵活的）选择器是jQuery的核心组成部分，因为使用jQuery操作DOM时所做的每件事都和选择器密切相关——总得先选取元素才可进行下一步。jQuery使用常见的CSS选择器和XPATH选择器，它们为绝大多数Web设计师和开发者所熟悉。除此之外，还有一些jQuery自定义的选择器。正是这些选择器使得jQuery分外灵活，易于学习。理解选择器如何工作，才能为充分利用jQuery的强大功能打好坚实基础。

在那些CSS选择器力不从心的场合，过滤器可以让你更灵活地根据DOM特性选取元素。人们常常结合使用过滤器和选择器，以便在基于某一标准选择特定元素时进行深度控制，比如需要根据元素在一组元素中的位置，或元素的可见性，或表单元素的某些属性（如选中/未选中或是否被禁用）选取元素时。jQuery还提供了一系列为DOM元素添加、删除CSS类或直接设定样式的方法。

在本章中，我将通过取自真实项目的解决方案详细讲解选择器、过滤器及CSS的使用方法，让大家彻底弄懂这些jQuery基础方法。

3.1 使用 jQuery 选择器选取 DOM 元素

选择器是jQuery库的一项基础功能，由jQuery Sizzle选择器引擎驱动。Sizzle引擎也能用于其他语言，却只有jQuery将它的威力发挥到了极致。对那些非常了解HTML与CSS的Web设计师来说，Sizzle的语法非常易于理解。jQuery Sizzle是用JavaScript语言写就的，专门用来处理jQuery选择器的一个引擎，它除了支持通用的CSS选择器和XPATH选择器，还支持一些jQuery自定义选择器。

jQuery选择器是一个字符串表达式，它负责选取（又叫匹配）一个或一组DOM元素，为下一步jQuery操作做准备。选择器总是在jQuery别名（$）之后声明。匹配并处理完DOM元素之后，匹配的结果集便成为一个jQuery对象，这样就可以在匹配元素上应用jQuery丰富的方法（如事件、特效、遍历、操作等）了。jQuery对象是这样一个东西，在编写jQuery代码的过程中，不管你有没有刻意留意过它，它总在那里，不离不弃。它很重要，应该知道它，重视它，了解它。

下面这行代码展示了在jQuery中选择器的用法：

```
$(selector).method();
```

不需要做太多练习，在jQuery中使用选择器就会成为一项本能，这是因为许许多多的选择器和CSS选择器完全相同。选择器是在DOM中穿行的最好工具，并且它选取元素的最基本形式与CSS选择器语法完全相同，不外乎ID、类、标签或属性这几种。在每一条jQuery语句中，选择器都是一个基本组件。

在使用选择器时，jQuery语句自动遍历DOM的全部节点，寻找与该选择器匹配的元素，遍历得到的结果又称为匹配集。那些精通CSS并透彻理解DOM的Web设计师可轻易上手jQuery选择器。

组成选择器的JavaScript部分有。

- jQuery别名（`jQuery`或`$`）。
- 要查找的DOM元素，被两个小括号及其中的引号包裹。
- 紧接着选择器的是你要用到的jQuery方法。从为选中的元素设定CSS样式到让页面元素动起来，jQuery方法几乎无所不能。这些方法（即函数）完成具体的事情并可接受括在小括号中的参数。

表3-1解析了jQuery选择器语句。

表3-1　jQuery选择器语句解析

jQuery别名	选　择　器	jQuery方法或行为
`$ 或 jQuery`	`('div')`	`.css('border','1px solid #333');`

使用CSS选择器选取页面元素

JavaScript本身提供了根据元素ID或标签选取元素的方法。这些方法的不便之处在于，你不得不为处理3种不同类型的元素各写一个函数。不用说，这会带来重复而臃肿的代码，而且会越来越难以管理和维护。jQuery的一个选择器可处理多种类型的元素，自然容易写出干净且便于管理的代码。

本书所有的例子都同时配有浏览器输出结果和Firebug输出结果，这样可方便我们查看DOM到底发生了哪些改变。（要了解如何安装和使用Firebug，请参阅第2章。）我们查看页面源代码的时候，没有办法看到经过JavaScript处理后的源代码，而JavaScript处理之后的代码才是页面真正对应的代码。（浏览器加载页面之后，我们可以使用JavaScript操作DOM，对页面做相当大的修改。）Firebug不但让我们看到修改之后的代码，甚至还能让我们看到源代码的实时变更过程，实在是一个测试JavaScript和jQuery的极佳工具。

下面列出了最常用的CSS选择器。接下来我将利用示例代码和浏览器深入介绍它们的用法：

- `$('*')`
- `$('p')`
- `$('.class')`
- `$('#id')`
- `$('.parent ul li')`

在本章中，我使用jQuery的`.css()`方法设置元素的样式。jQuery的CSS方法支持任意的CSS属性参数，并将相应的样式应用到选择器匹配的元素上。这些CSS属性将在DOM完全加载后以内嵌样式的方式应用到目标元素上。

这些例子中，你不必拘泥于我使用的CSS属性。我只是演示几个你可能会遇到的问题，教你用选择器迅速解决它们。

1. 使用通配符选择器选取元素

如果你打算选取DOM中所有的元素，或者某个元素内的所有元素，那就用通配符选择器(＊)。在jQuery别名$符号之后，用一对小括号把用引号包括的星号（＊）包起来，就构成了通配符选择器。

在下面的HTML示例里，我使用通配符选择器选取页面中所有的元素，并为每个元素设置CSS边框。我使用下面的语句设置边框样式：

```
$('*').css('border', '1px solid #333');
```

图3-1展示了浏览器中下面这个脚本示例的输出，图中Firebug已激活，且打开了HTML标签，这样你就能方便地观察到jQuery为页面中的每个元素添加内嵌边框样式的过程。

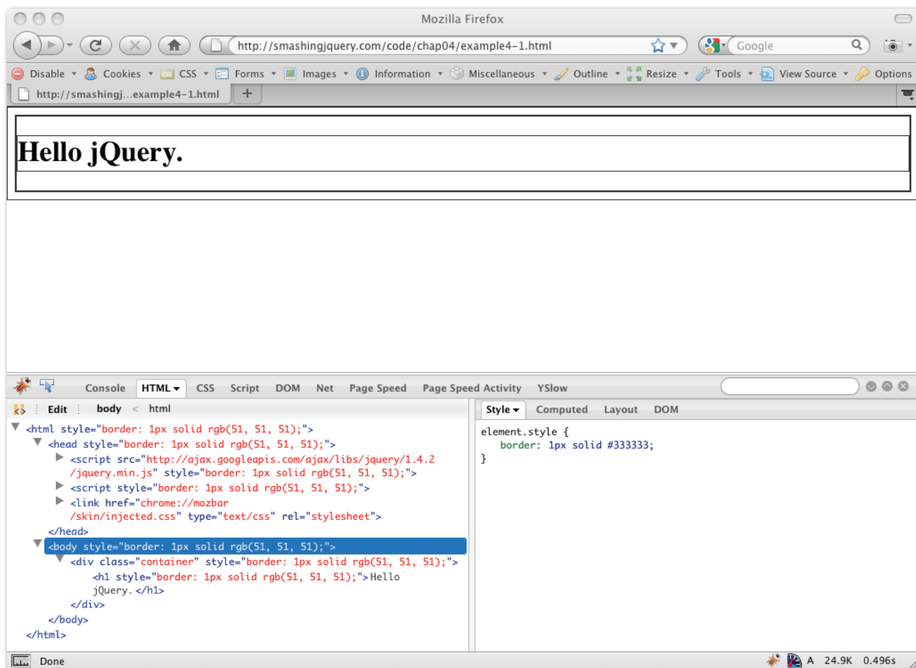


图3-1　浏览器输出（使用通配符选择器为页面上的每个元素添加了边框）

```
<!doctype html>
<html>
  <head>
```

```
    <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'>
      </script>
    <script>
    $(document).ready(function() {
      $('*').css('border','1px solid #333');
    });
    </script>
    <body>
      <div class="container">
      <h1>Hello jQuery.</h1>
      </div>
    </body>
  </html>
```


2. 使用HTML标签选择元素

在了解通配符选择器如何工作之后，接下来我们会看到其他CSS选择器也是以同样的方式工作。使用元素选择器——把标签名作为参数传递给选择器，我们就可以选中DOM中（页面内）该标签对应的所有元素。这个选择器内部使用的是原生JavaScript方法`getElementsByTagName()`。

原生的JavaScript方法`getElementsByTagName()`能根据标签名字获取标签对应的所有元素，相应代码如下：

```
document.getElementsByTagName('h1');
```

在下面这个例子中，我将设定`h1`标签的`font-family`属性。我可以通过修改级联样式表达到目的，但并不想影响其他页面的`h1`标签（修改CSS则会）。因此，我使用标签选择器选中元素，然后用`css()`方法改变当前页面中标签为`h1`的元素的`font-family`属性。这里的CSS方法只会作用于我选中的`h1`元素。

图3-2展示了浏览器中下面这个脚本示例的输出，图中Firebug已激活，且打开了HTML标签。这个例子演示了DOM加载完成之后，DOM中`h1`元素的`font-family`属性被修改的过程。

```
<!doctype html>
<html>
  <head>
  <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'>
  </script>
  <script>
  $(document).ready(function() {
    $('h1').css('font-family','arial,verdana');
  });
  </script>

  <body>
    <div class='container'>
    <h1>Hello jQuery.</h1>
    </div>
  </body>
</html>
```

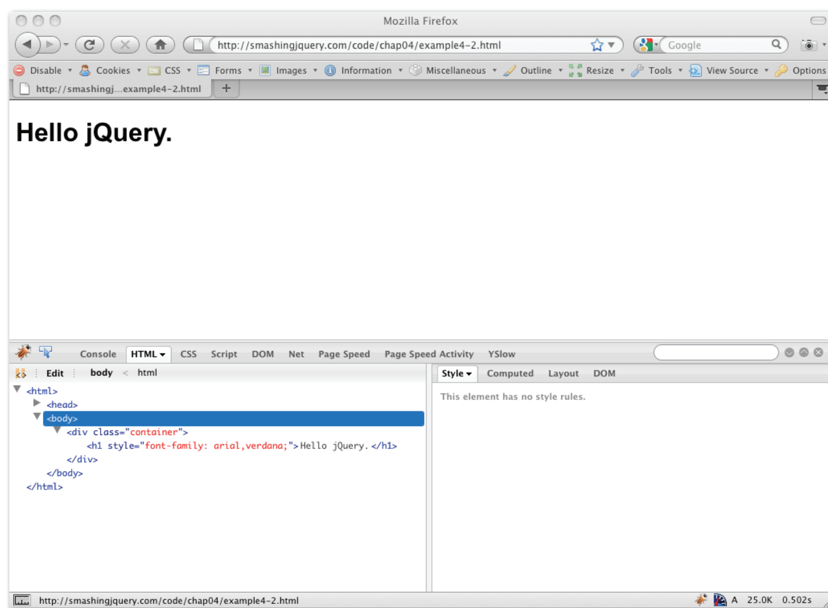


图3-2　工作中的元素选择器

3. 使用ID选择器选取元素

使用ID('#')选择器可选中页面中任意一个ID对应的元素。ID选择器使用的是原生JavaScript方法getElementById()，代码如下：

```
document.getElementById('sidebar');
```

ID选择器总是以#字符开头，如果忘记书写这个符号，就无法选中正确的元素。若ID选择器匹配成功，就会返回匹配的唯一元素。我们知道，在一个页面中一个ID只能对应一个元素。

在下面的例子中，我打算使用CSS隐藏#sidebar这个div。我使用jQuery选择器$('#sidebar')把它从文档中选出来。为了隐藏这个div元素，我使用CSS方法把该元素的display属性设置为none。

图3-3展示了使用ID选择器隐藏#sidebar 对应div元素的浏览器输出：

```
<!doctype html>
<html>
  <head>
  <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'>
  </script>
  <script>
  $(document).ready(function() {
    $('#sidebar').css('display','none');
  });
  </script>
  <body>
  <div id='sidebar'>
```

```
    <h1>My sidebar</h1>
    <ul>
      <li><a href='/nav'>Navigation</a></li>
    </ul>
    </div>
    </body>
  </html>
```

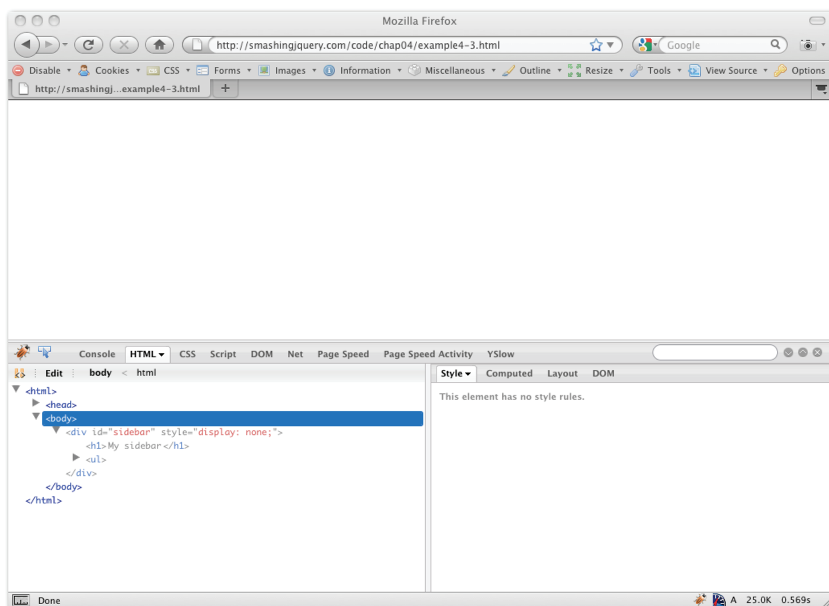



图3-3　选中#sidebar元素后的浏览器输出

4. 使用类选择器选取元素

在页面中我们可以用ID选取元素，类似地，我们也可以用类（.class）选取元素。这个选择器使用JavaScript原生方法getElementsByClassName()完成工作。[①]类选择器用来选择DOM中具有特定类的所有元素。如果我们用原生JavaScript方法getElementsByClassName选取一个类，写出来的代码就像下面这样：

```
document.getElementsByClassName('product-image');
```

使用jQuery，我们可以使用非常少的代码实现getElementsByClassName()方法的目标。

在下面的例子中，我打算为（图片）设置1 px宽的灰色边框，5 px的内边距，并将它的宽度设置为150 px。jQuery的CSS方法支持对象字面量（一个逗号分隔的名/值对列表，有助于组织代码）参数，以这种方式一次设置多个CSS属性，代码既简单又整洁。

下面我使用类选择器选中页面中所有的.telephone元素，并传给CSS方法3个CSS属性，

① 较老的浏览器没有getElementsByClassName方法，对这些浏览器，jQuery使用自己的算法选取元素。

因此我需要用一对大括号将这些属性括起来。图3-4展示的是选中元素并设置样式之后的浏览器输出。

```
<!doctype html>
<html>
  <head>
  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js">
  </script>
  <script>
  $(document).ready(function() {
    $('.telephone').css({'padding':'5px','border':'1px solid #ccc','width':'150px'});
  });
  </script>
  <body>
    <div id='container'>
    <h1>Hello jQuery</h1>
    <div class='telephone'><img src="images/product.jpg"></div>
    <div class='telephone'><img src="images/product.jpg"></div>
    <div class='telephone'><img src="images/product.jpg"></div>
    <div class='telephone'><img src="images/product.jpg"></div>
    <div class='telephone'><img src="images/product.jpg"></div>
    <div class='telephone'><img src="images/product.jpg"></div>
    <div class='telephone'><img src="images/product.jpg"></div>
    <div class='telephone'><img src="images/product.jpg"></div>
    </div>
  </body>
</html>
```

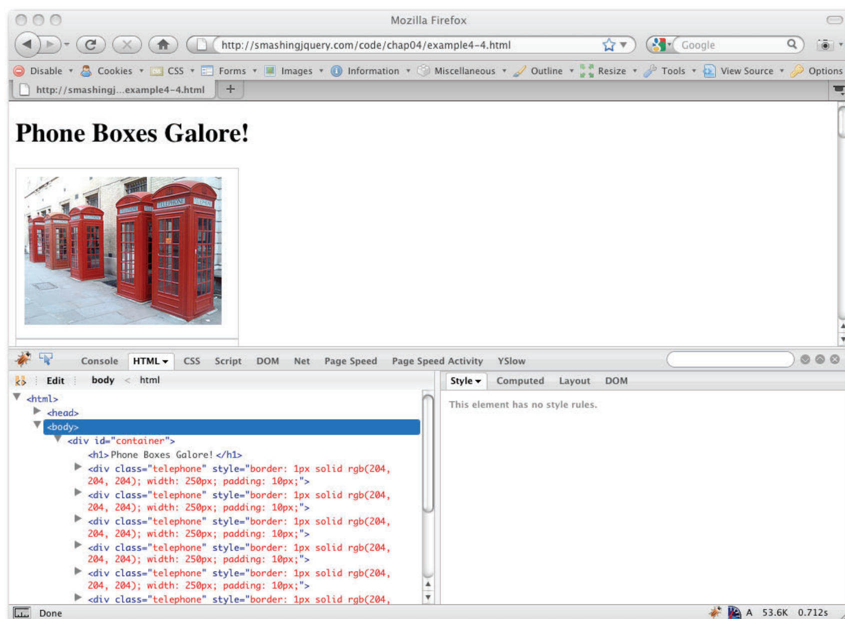


图3-4　选中页面中拥有telephone类名的元素并设置样式之后的浏览器输出

5. 使用多个类选取一个或多个元素

在某些场合，我们会给一些元素设置多个类，并打算只选取这些同时拥有多个类的元素。在类选择器中可以使用多个类。

下面的例子里，有6个元素具有不止1个类。我打算使用jQuery的类选择器和CSS方法隐藏那些同时具有book类和inactive类的元素。图3-5展示的是选择器匹配的多个元素成功被隐藏之后的浏览器输出。

```
<!doctype html>
<html>
  <head>
  <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'>
    </script>
  <script>
  $(document).ready(function() {
    $('.book.inactive').css('display','none');
  });
  </script>

  <body>
    <div class='book inactive'>
        <p>Travel Guide to NYC</p>
    </div>
    <div class='book active'>
        <p>Travel Guide to San Francisco</p>
    </div>
    <div class='book inactive'>
        <p>Travel Guide to Seattle</p>
    </div>
    <div class='book active'>
        <p>Travel Guide to Miami</p>
    </div>
    <div class='book active'>
        <p>Travel Guide to Palo Alto</p>
    </div>

  </body>
</html>
```

6. 使用子元素选择器选取元素

当无法使用标签、CSS和ID选择器时，我们可以使用子元素选择器这种非常有用的页面元素选择器。除了IE6，其他现代浏览器都支持CSS中的子元素选择器。你惊讶吗？我不惊讶[①]。最棒的是jQuery中的子元素选择器支持IE6。在嵌套元素（如导航菜单）中选取元素时，子元素选择器特别有用。

子元素选择器通过 > 选择符选取给定父元素的直接子元素。看看下面这个语句，它选取body元素内的所有直接子元素p：

```
$('body > p')
```

① 任何有经验的Web开发者都不会惊讶。.

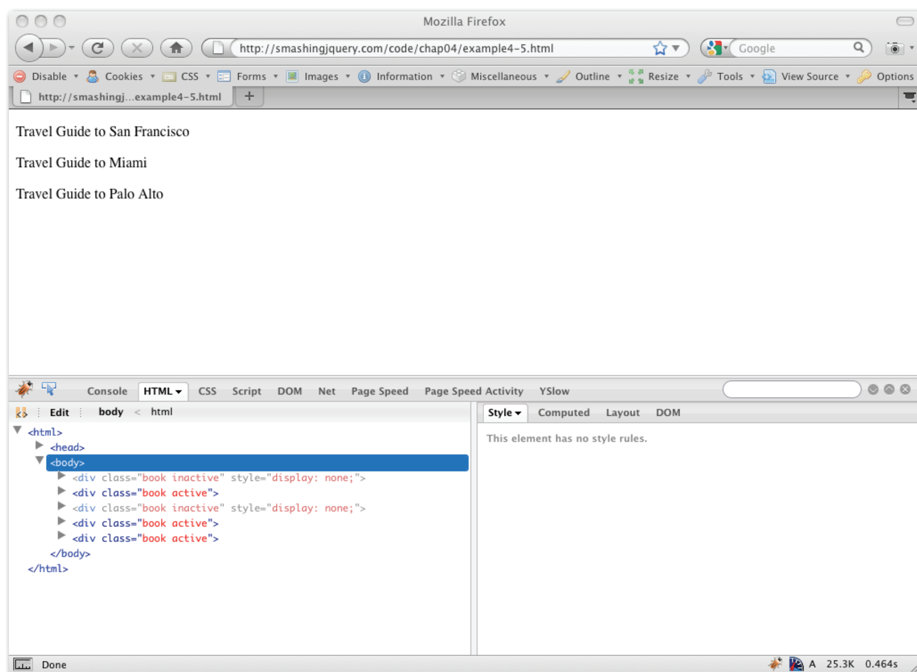


图3-5　选取同时具有book和inactive类的两个元素并隐藏它们之后的浏览器输出

不过，如果需要在某个div元素内选取某个特殊的子元素p，为了找出正确的后代元素，我们就不得不在选择器中加上该元素更具体的类名或ID。

在下面这个例子里，我先找出具有.inactive类的元素，再选取它的子元素p，然后在该段落的末尾用红字加上“Sorry, this book is sold out”字样。我会使用链式调用方法把这一连串动作写在一个语句中。图3-6展示了该语句执行后的浏览器输出。

```
<!doctype html>
<html>
  <head>
  <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'>
  </script>
  <script>
  $(document).ready(function() {
    $('.book.inactive > p').css('display','none');
    $('.book.inactive').append('Sorry this book is sold out!').css('color',
    'red');
  });
  </script>

  <body>
    <div class='book inactive'>
        <p>Travel Guide to NYC</p>
    </div>
```

```
    <div class='book active'>
        <p>Travel Guide to San Francisco</p>
    </div>
    <div class='book inactive'>
        <p>Travel Guide to Seattle</p>
    </div>
    <div class='book active'>
        <p>Travel Guide to Miami</p>
    </div>
    <div class='book active'>
        <p>Travel Guide to Palo Alto</p>
    </div>

  </body>
</html>
```

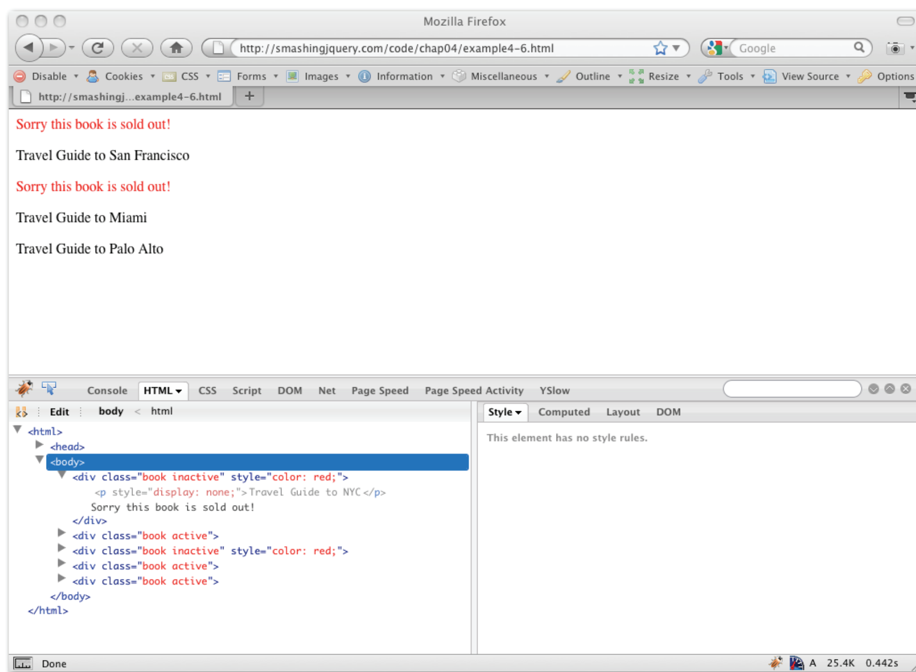



图3-6 选取元素并设置样式之后的浏览器输出（另见彩插图3-6）

7. 使用后代元素选择器选取元素

子元素选择器只匹配直接子元素，比如匹配ul标签内的li标签。如果要选取目标元素的孙元素、重孙元素等，就需要使用后代元素选择器。子元素选择器和后代元素选择器的区别在于是否使用 > 操作符。如果去掉子元素选择器中的 > 符号，就能够选取所有的后代元素（包括子元素）。

在下面这个例子中，我先找出<ul class="sidebar-nav">标签内的所有li标签，然后再为它们添加边框样式。图3-7展示了选取后代元素并设置样式之后的浏览器输出。

```
<!doctype html>
<html>
  <head>
  <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'>
  </script>
  <script>
  $(document).ready(function() {
    $('ul li').css('border','3px dashed blue');
  });
  </script>

  <body>
    <ul class="sidebar-nav">
    <li>Link 1</li>
    <li>Link 2</li>
    <li><ul>
        <li>Sub Link 1</li>
        <li>Sub Link 2</li>
    </li>
    </ul>
  </body>
</html>
```

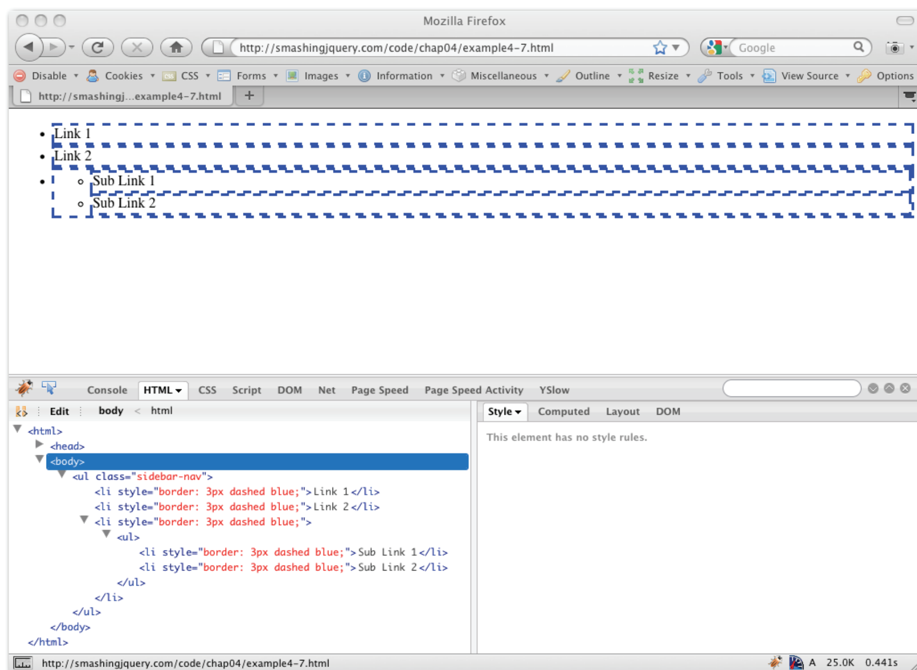


图3-7　选取后代元素并设置样式之后的浏览器输出（另见彩插图3-7）

8. 组合选择器

有时候我们需要同时选取多种类型的元素，这就需要组合使用类选择器、ID选择器、标签选择器或者子元素选择器。在jQuery中，使用逗号分隔的选择器列表字符串就可达成目的。

在下面这个例子中，我先用逗号分隔开5个不同的类和2个不同的ID，然后使用CSS方法把所有匹配元素的背景设置成灰色。图3-8展示了选取元素并设置样式之后的浏览器输出。

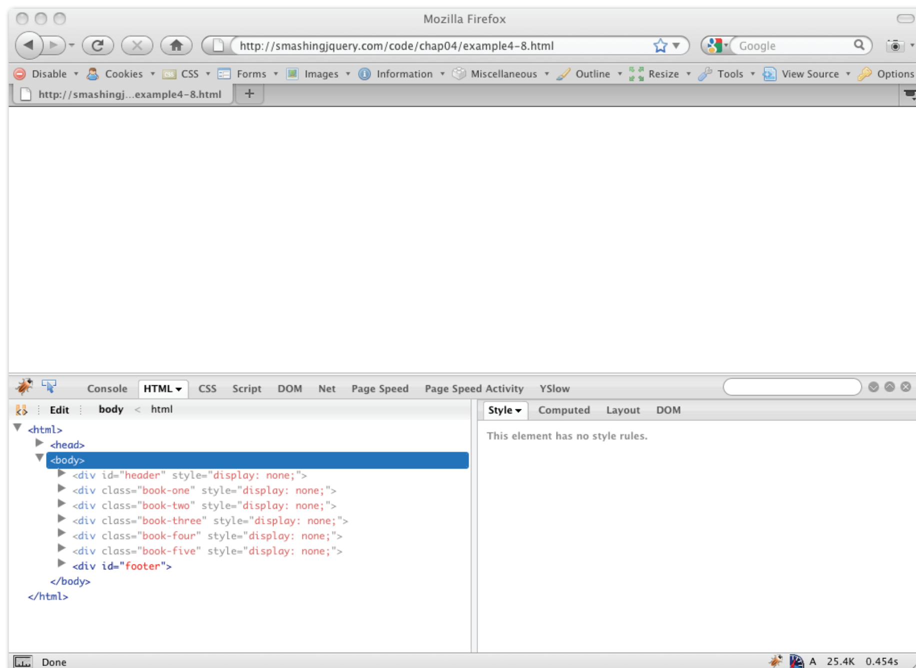


图3-8　选取多种类型元素并设置样式之后的浏览器输出

```
<!doctype html>
<html>
  <head>
  <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'>
  </script>
  <script>
  $(document).ready(function() {
    $('.book-one, .book-two, .book-three, .book-four, .book-five, #header,
      #footer p').css('background','#ccc');
  });
  </script>

  <body>
    <div id='header'><h1>Book Club</h1></div>
    <div class='book-one'>
        <p>Travel Guide to NYC</p>
    </div>
```

```
        <div class='book-two'>
            <p>Travel Guide to San Francisco</p>
        </div>
        <div class='book-three'>
            <p>Travel Guide to Seattle</p>
        </div>
        <div class='book-four'>
            <p>Travel Guide to Miami</p>
        </div>
        <div class='book-five'>
            <p>Travel Guide to Palo Alto</p>
        </div>
        <div id='footer'><p>Copyright 2010</p></div>
    </body>
</html>
```

3.2　使用 jQuery 过滤器过滤元素

过滤器用来进一步提炼选择器匹配的元素。当我们只需要在DOM中选取一个或几个元素时，过滤器非常好用。如果只是处理一个静态HTML文档，调整HTML非常简单。但是，如每一次页面请求或加载时，DOM都会变动，我们就需要使用前端语言（如JavaScript）动态添加格式。

过滤器的格式为一个冒号加上过滤器的名字，即`:过滤器名`。

3.2.1　基本过滤器及应用

自jQuery 1.4.2起，jQuery一共定义了20个过滤器。我会给出最常用过滤器的真实应用示例，所有可用的jQuery过滤器见表3-2。

在表3-2中列出的过滤器当中，CSS选择器一栏打着“是”标志的伪类是在css3标准中定义的。目前只有比较新的浏览器完整地支持这些属性，如Firefox 3+、Opera 10+、Safari 3+。若需要查看完整的CSS属性列表，可参考www.css3.info网站。jQuery完全支持这些CSS3属性，它们可以安全的作为过滤器结合选择器使用。

表3-2　过滤器及其功能列表

名　　字	功　　能	属于CSS3选择器
:even 和 :odd	匹配结果集中顺序为偶数（:even）或奇数（:odd）的元素	
:header	匹配标题元素H1、H2、H3、H4等标签	
:not	不匹配后面选择器的元素	是
:eq(index)	匹配顺序号等于index的元素	
:gt(index)	匹配顺序号大于index的元素	

（续）

名　字	功　能	属于CSS3选择器
:lt(index)	匹配顺序号小于index[1]的元素	
:first-child、:last-child、:only-child、nth-child(index)[2]	分别匹配第一个、最后一个、“独生子”元素（有父元素但无兄弟元素）、第index个子元素	是
:has(p)	匹配包含p元素的元素	
:contains('this is my text')	匹配包含this is my text文本[3]的元素	
:empty	匹配无内容的元素（如<img />或 ）	是
:parent	匹配拥有子元素的元素	
:hidden	匹配不可见的元素	
:visible	匹配可见的元素	
:animated	匹配正处于动画过程中的元素	

3.2.2 利用**:even**和**:odd**过滤器生成条纹表格

为表格的奇数行或偶数行加上淡淡的灰色背景（条纹效果）从而使表格更容易阅读，是很常见的做法。jQuery中的:even和:odd过滤器让这件事情做起来分外容易。这两个过滤器并非为实现条纹效果而生，利用它们还能做其他很多很多事情，不过话说回来，制作条纹效果确实是一个很好的教学示例。图3-9展示了页面载入完成后为表格偶数行添加背景色之后的浏览器效果。

```
<!doctype html>
<html>
  <head>
  <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'>
  </script>
  <script>
  $(document).ready(function() {
    $('tr:even').css('background','#dedede');
    $('tr:odd').css('background','#ffffff');
  });
  </script>

  <body>
    <table>
      <tr>
      <th>Product</th>
```

① :eq(incex)、:gt(index)、:lt(index)中的index是从0开始计数的。
② 这里的index从1开始计数。
③ 此处匹配文本区分大小写，可以加引号，也可以不加。

```
        <th>Description</th>
        <th>Price</th>
        </tr>
        <tr>
        <td>Paper Towels</td>
        <td>The most absorbent paper towels.</td>
        <td>$18.99</td>
        </tr>
        <tr>
        <td>Paper Napkins</td>
        <td>Perfect for your outdoor gathering.</td>
        <td>$16.99</td>
        </tr>
        <tr>
        <td>Paper Plates</td>
        <td>The best value.</td>
        <td>$5.99</td>
        </tr>
        <tr>
        <td>Plastic Forks</td>
        <td>The essential picnic accessory.</td>
        <td>$2.99</td>
        </tr>
      </table>
    </body>
  </html>
```

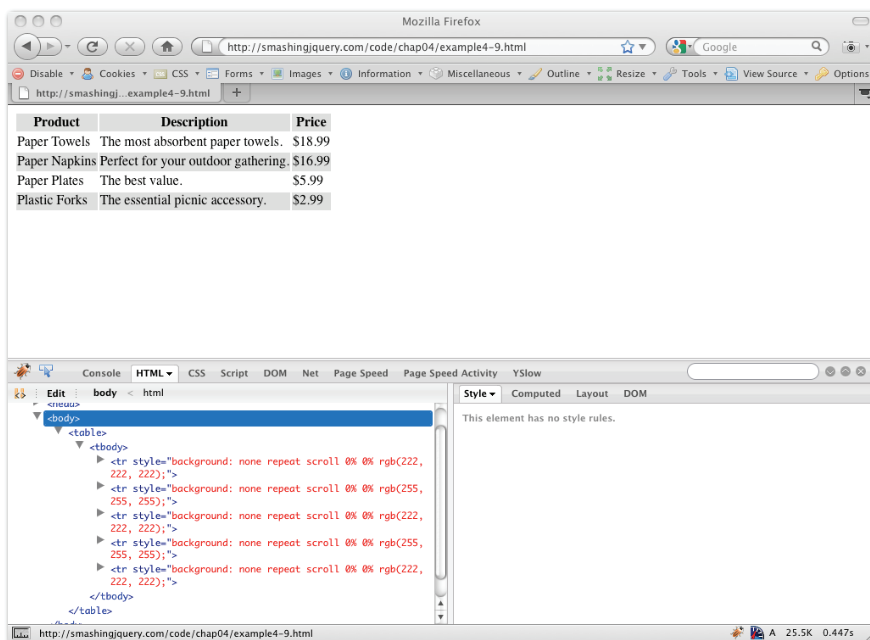


图3-9　表格的偶数行背景与其他行不同

3.2.3　为列表或集合中的第一个和最后一个元素设置样式

如果需要从DOM内的某个元素集中找出第一个或最后一个元素，当然是使用`:first`和`:last`过滤器最简便。这两个过滤器依据结果集的索引（至多）返回一个元素。

在下面这个例子中，我会为列表中除第一个和最后一个之外的每个元素添加下边线。有两种方法可以达到这个目标，一种方法是为列表中的最后一个（和第一个）元素添加一个`.last`类，另一种方法是结合使用`:first`和`:last`过滤器及`ul li`选择器。图3-10展示了在这个列表上应用`:first`和`:last`过滤器之后的效果。

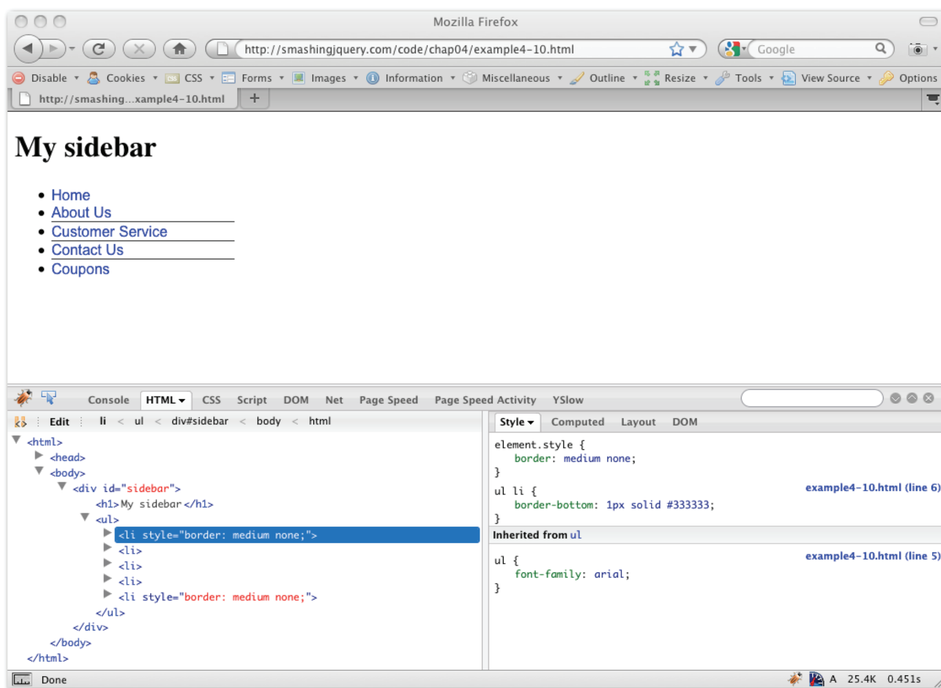


图3-10　除了第一个和最后一个，列表中所有元素都有下边线

```
<doctype html>
<html>
  <head>
  <style>
  ul {width:200px;font-family:arial;}
  ul li {border-bottom:1px solid #333;}
  ul li a {text-decoration:none;}
  </style>
  <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'>
  </script>
  <script>
  $(document).ready(function() {
```

```
    $('ul li:first').css('border','none');
    $('ul li:last').css('border','none');
  });
  </script>
  <body>
    <div id='sidebar'>
    <h1>My sidebar</h1>
    <ul>
      <li><a href='/index'>Home</a></li>
      <li><a href='/about'>About Us</a></li>
      <li><a href='/customer-service'>Customer Service</a></li>
      <li><a href='/contact'>Contact Us</a></li>
      <li><a href='/coupons'>Coupons</a></li>
    </ul>
    </div>
  </body>
</html>
```

3.2.4　找出包含特定元素的元素

有时候，我们需要找出那些包含某个元素的元素。在这种情况下，:has过滤器就派上了用场。:has并不要求被包含的元素是父元素的直接子元素，只要是后代元素就行。

下面这个例子中，我使用:has过滤器找出具有.content类并且包含p标签的元素，把它的字号设置为18 px。图3-11展示了浏览器的显示效果。

```
<!doctype html>
<html>
  <head>
  <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'>
  </script>
  <script>
  $(document).ready(function() {
    $('.content:has(p)').css('font-size''','''18px');
  });
  </script>

  <body>
    <div id="main">
      <div class="content">
      <p>This is my content</p>
      </div>
      <div class="alternate">
      <p>This is alternate content.</p>
      </div>
    </div>
  </body>
</html>
```

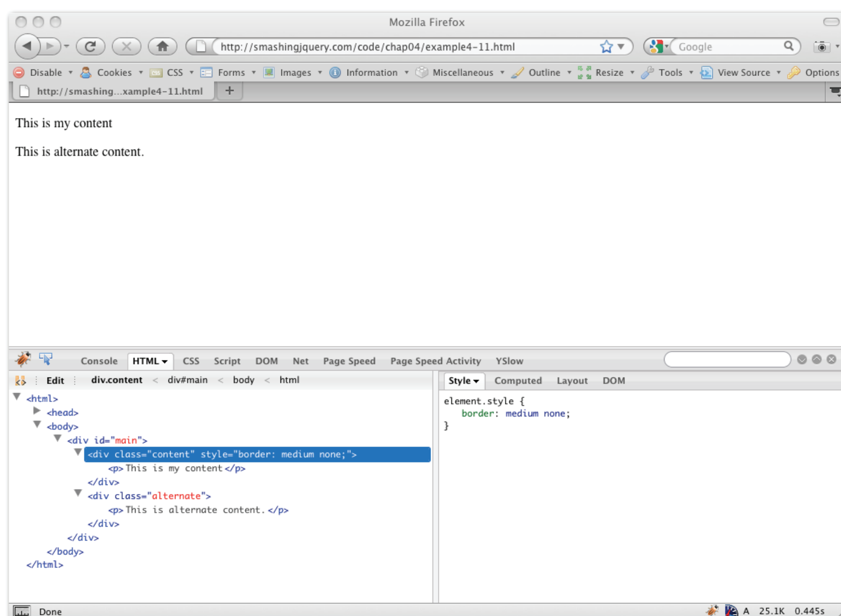


图3-11　包含p标签的.content元素

3.2.5　找出不包含任何子元素或文本的元素

如果要找出那些空无一物的元素（不包含子元素，也不包含文本），:empty过滤器就很给力。

在下面这个例子中，如果.error div为空，我就把它藏起来。这很好办，只要使用.error这个类选择器加一个:empty伪类过滤器就可以了。如果.error元素有内容，它照常显示，否则设置它的display属性为none。图3-12展示的是这个选择器匹配成功的效果。

```
<!doctype html>
<html>
  <head>
  <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'>
  </script>
  <script>
  $(document).ready(function() {
    $('.error:empty').css('display','none');
  });
  </script>

  <body>
    <div id="main">
      <div class="error"></div>
      <div class="error">This is my error message</div>
      <div class="content">
      <p>This is my content</p>
      </div>
      <div class="alternate">
```

```
        <p>This is alternate content.</p>
        </div>
      </div>
    </body>
  </html>
```

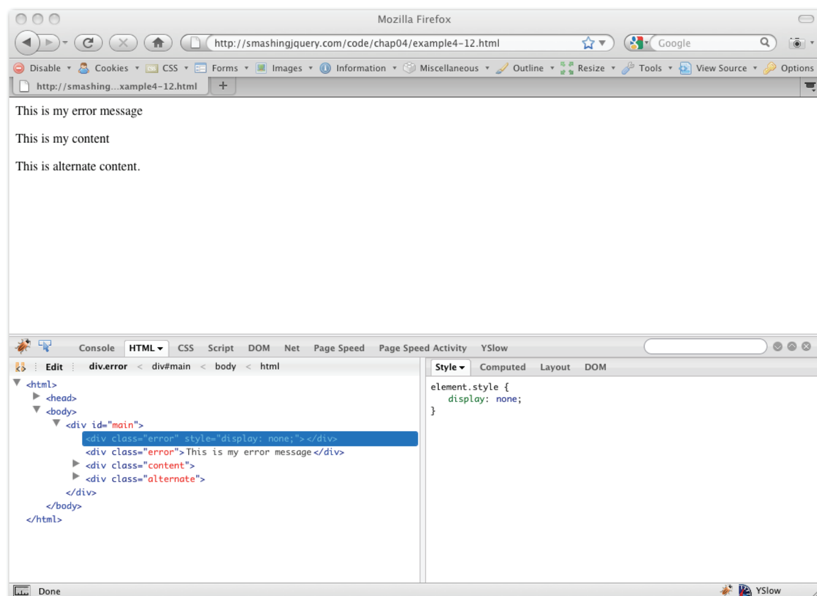


图3-12　选择器成功匹配，.error元素被隐藏起来

3.2.6　根据元素包含的文本过滤元素

有时候需要根据元素包含的内容匹配元素，我们可以用:contains过滤器实现这一目标。传递给:contains过滤器的文本可以用引号括起来，也可以不用。

在下面的例子中，我将使用:contains过滤器找出表格中包含Paper Towels字样的单元格，为其添加1 px虚线边框。图3-13展示的是这个选择器匹配成功并设置样式后的效果。

```
<!doctype html>
<html>
  <head>
  <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'>
  </script>
  <script>
  $(document).ready(function() {
    $("tr td:contains('Paper Towels')").css('border','1px dashed #333');
  });
  </script>

  <body>
    <table>
```

```
      <tr>
      <th>Product</th>
      <th>Description</th>
      <th>Price</th>
      </tr>
      <tr>
      <td>Paper Towels</td>
      <td>The most absorbent paper towels.</td>
      <td>$18.99</td>
      </tr>
      <tr>
      <td>Paper Napkins</th>
      <td>Perfect for your outdoor gathering.</th>
      <td>$16.99</th>
      </tr>
      <tr>
      <td>Paper Plates</td>
      <td>The best value.</td>
      <td>$5.99</td>
      </tr>
      <tr>
      <td>Plastic Forks</td>
      <td>The essential picnic accessory.</td>
      <td>$2.99</td>
      </tr>
    </table>
  </body>
</html>
```

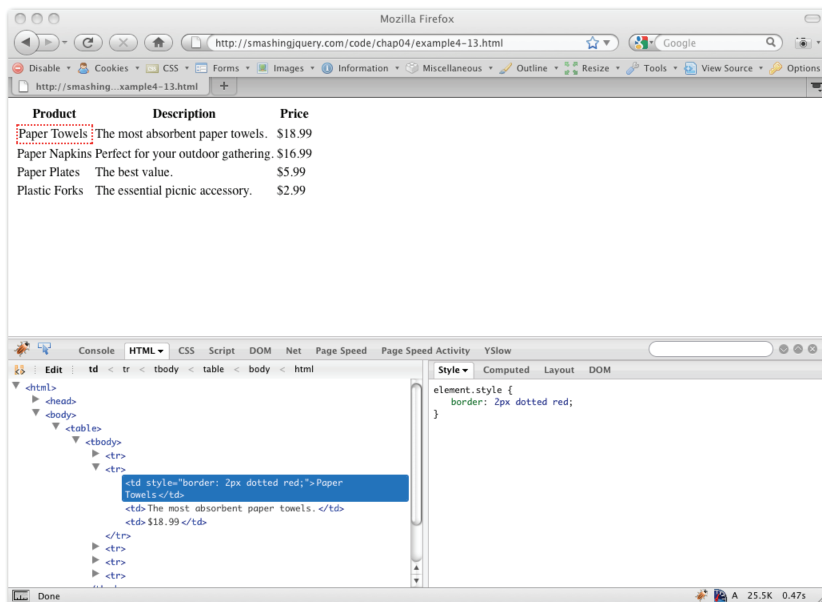



图3-13　选择器匹配包含“Paper Towels”字符串的td成功并设置样式后的浏览器输出

3.3　根据元素的属性在 DOM 中选取元素

标签属性用来记录标签的附加信息，或者用于和JavaScript交互。比如很常见的HTML标签<a>，它就可以有href、rel、id、class、title、hreflang等属性。在表单中通过属性选取表单项是很好的做法，因为我们可以通过名称、类型、属性、类、ID等检索元素。

表3-3中的各种属性选择器均可用于从DOM中选取元素。本书后面会通过一个较复杂的案例详细讲解属性，现在我准备了两个常见的例子来说明属性选择器的用法。

表3-3　属性名字及其功能

属　性　名	属性功能
`$('[attribute*=value]')`	若value是目标属性的值的一部分，则匹配成功（*表示通配）
`$('[attribute|=value]')`	目标属性的值等于value或者以value开头且其后紧跟一个连字符（-）
`$('[attribute~=value]')`	目标属性的值包含value中的任意一个子串（value是这样一个字符串，它由多个子串组成，子串之间由空格分隔）
`$('[attribute$=value]')`	目标属性的值以value结束
`$('[attribute=value]')`	目标属性的值等于value
`$('[attribute^=value]')`	目标属性的值以value开始
`$('[attribute!=value]')`	目标属性的值不等于value
`$('[attribute=value][attribute=value][attribute=value]')`	目标元素同时匹配多个属性及其值

3.3.1　选择包含某个网站地址的链接

通配符*是一个灵活又常用的属性选择器，用来在整个DOM中搜索包含特定属性值的元素。这个选择器非常快：如果你打算用一双肉眼找出这些元素，怕要花上几个小时，而使用通配符属性选择器，得到同样的结果只需要几秒钟。

在下面这个例子中，我要在所有的链接中找出href属性中包含google.com域名的那些，然后给它们添加一个CSS类（为链接加上一个小型谷歌图标）。我先找出那些href中包含google.com的链接，然后调用addClass()方法添加.google-icon类。图3-14展示的是这个选择器匹配成功并设置样式后的效果。

```
<!doctype html>
<html>
  <head>
  <style>
    .google-icon {background:url(images/google_icon.jpg) no-repeat;padding:0px
                  30px;}
  </style>
  <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'>
```

```
    </script>
    <script>
        $(document).ready(function () {
            $('ul li a[href*="google.com"]').addClass('google-icon');
        });
    </script>

    <body>
      <ul>
        <li><a href=http://www.google.com/analytics>Google Analytics</a></li>
        <li><a href="http://www.yahoo.com/sports">Yahoo Sports</a></li>
      </ul>
    </body>
</html>
```

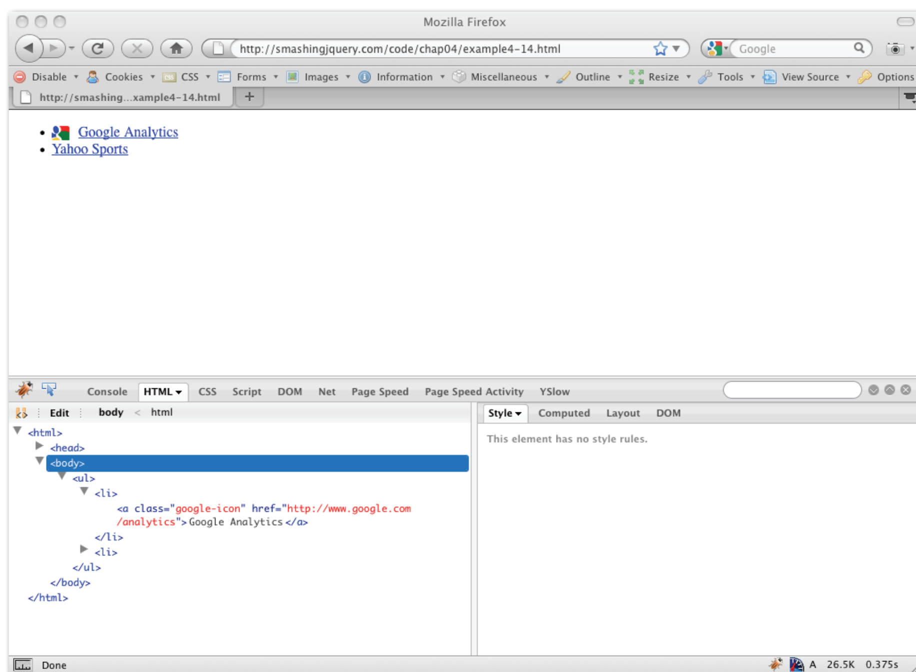



图3-14　匹配href中包含google.com域的链接并设置样式后的浏览器输出

3.3.2　选择属性值以某个单词结尾的元素

类似于在属性过滤中使用通配符，我们可以用美元符号（$）选取属性值以某个字符串结尾的元素。在下面这个例子中，我有4个div元素，这4个元素的ID属性都不相同，但我想为它们加上同一个类。好在这里所有div元素的id均以bird这个单词结尾，所以我可以用属性选择器利用这个单词选取元素。图3-15展示的是匹配bird成功并设定样式之后的浏览器输出。

```
<!doctype html>
<html>
  <head>
  <style>
  .bird {border:1px solid #ccc;width:200px;margin:5px;}
  </style>
  <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'>
  </script>
  <script>
      $(document).ready(function () {
          $('div[id$="bird"]').addClass('bird');
      });
  </script>
  <body>
    <div id="red-bird"></div>
    <div id="blue-bird"></div>
    <div id="green-bird"></div>
    <div id="black-bird"></div>
  </body>
</html>
```

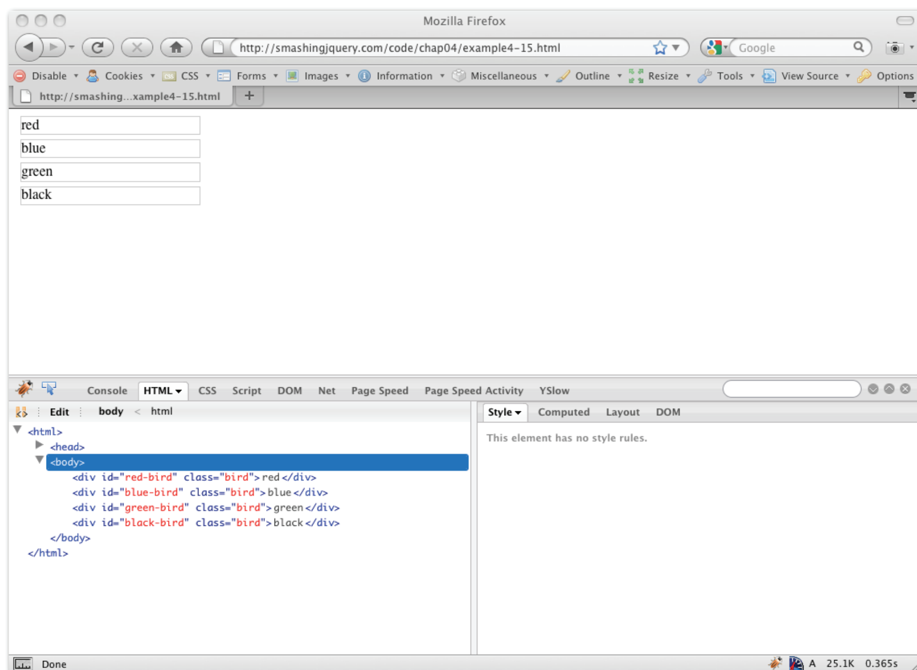


图3-15　匹配bird成功并设置样式之后的浏览器输出

3.3.3 用jQuery操作HTML和CSS

学会使用选择器在DOM中漫游是成功理解jQuery的第一步。不过，选取一个元素之后，总要对它们做点什么，比如添加、删除、克隆、替换或者设置样式。由于我可以假定选择器会返回一个元素，在后面我会把选中的元素称为结果集。

使用jQuery的乐趣自操作DOM开始。我们已经了解了几种为DOM设置CSS属性的方法，接下来我们准备走得更远一点，了解如何真正地定制一个页面。

如果你可以熟练使用原生JavaScript函数操作DOM，就会惊讶地发现使用jQuery代码更少，却能做得更好并且其语法更容易理解。

类似于操作页面中的HTML元素，我们也能够操作页面内容。

3.3.4 添加、删除、克隆及替换DOM元素或内容

在DOM中添加、删除、替换、克隆元素及其内容是jQuery的拿手好戏。举个例子，在某个公司我需要改变页面中的某些HTML，只好找到做后端开发的某位程序员，由他先修改编译后端的代码，然后用户才能看到改变后的HTML。这要花很多时间，尤其不适合只需要做一点点修改的情况。这是JavaScript擅长的领域。在一个生产服务器上更新JavaScript文件要比上传页面模板、修改配置文件等方便得多。

我可以（独立地）使用jQuery和CSS操作DOM，这要比麻烦一个后端程序员快得多。下面这个例子展示了使用jQuery操作DOM的一些常用技术。

1. 向DOM中添加HTML

`.html()`方法内部使用了原生的`innerHTML`属性，它既能直接调用`.html()`方法获取某个元素内的HTML文本，又可以传递HTML代码参数给`.html()`方法以设定其内部的HTML。

你必须清楚地明白`.html()`方法抓取目标元素内的全部DOM元素。图3-16展示的是在DOM加载完成，`.html()`方法执行之后的浏览器输出。

```
<!doctype html>
<html>
  <head>
  <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'>
  </script>
  <script>
  $(document).ready(function() {
    $('.content').html('<div class="main">Hello jQuery.</div>');
  });
  </script>

  <body>
    <div class="content">
      <p>I run 4 times a week.</p>
      <p>I lift weights 3 times a week.</p>
    </div>
  </body>
</html>
```

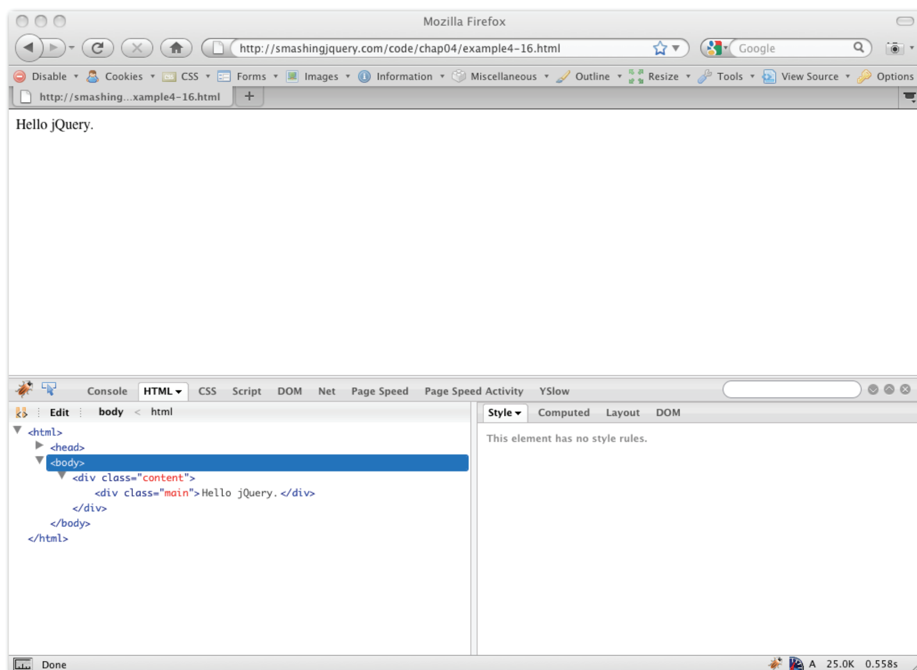


图3-16　DOM加载完成之后，.html()方法执行之后的浏览器输出

2. 设置DOM元素文本

.text()方法和.html()方法非常相似，只不过它仅抓取匹配元素内的字符串形式的文本。它的工作方式也和.html()方法相同：没有参数时获取元素文本，提供参数时设定元素文本。我们只能以字符串的形式传递文本参数给.text()，不能是HTML代码。

```
$('.content').text('Testing 1 2 3.');
```

3. 在一个元素内追加或在其最前面插入HTML

如果你打算在某个元素内添加一些HTML，可以使用.append()方法将它追加到目标元素内容之后，也可以用.prepend()方法将它插入到目标元素内容之前。.prepend()方法总是将参数插入到目标元素内第一个子元素之前，而.append()则总是将参数追加到目标元素内最后一个子元素之后。图3-17展示的是为.content元素追加内容之后浏览器的显示情况。

```
<!doctype html>
<html>
  <head>
  <style>
    .google-icon {background:url(images/google-icon.gif)no-repeat;}
  </style>
  <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'>
  </script>
  <script>
```

```
    $(document).ready(function() {
      $('.content').append('<p>I ride my bike 3 times a week.</p>');
    });
    </script>

    <body>
      <div class="content">
        <p>I run 4 times a week.</p>
        <p>I lift weights 3 times a week.</p>
      </div>
    </body>
  </html>
```

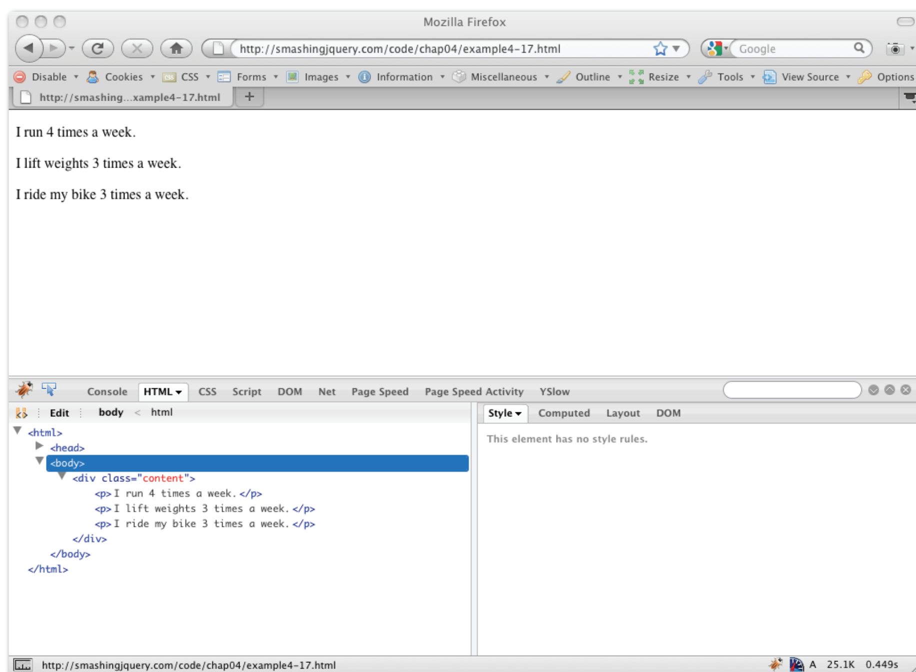



图3-17　为.content类对应的元素追加内容之后浏览器的输出

4. 在DOM某个元素之前或之后添加HTML

.before()和.after()方法与.append()和.prepend()方法非常相似。它们的区别主要在于HTML的插入位置。.before()和.after()并不把HTML插入到目标元素内部，而是将HTML插入到目标元素之前或之后。图3-18展示的是将HTML插入到.content元素之后浏览器的输出。

```
<!doctype html>
<html>
  <head>
  <style>
    .google-icon {background:url(images/google-icon.gif)no-repeat;}
```

```
    </style>
    <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'>
    </script>
    <script>
    $(document).ready(function() {
      $('.content').after('<p>I ride my bike 3 times a week.</p>');
    });
    </script>

    <body>
      <div class="content">
        <p>I run 4 times a week.</p>
        <p>I lift weights 3 times a week.</p>
      </div>
    </body>
  </html>
```

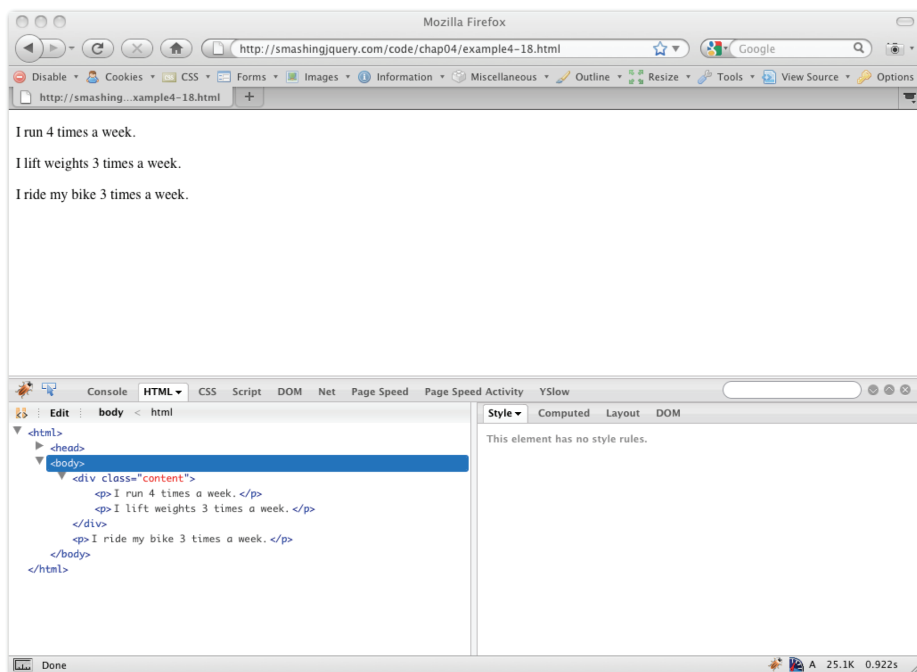


图3-18　把HTML插入到.content类对应元素之后浏览器的输出

5. 从DOM中删除HTML元素

.remove()方法用来从DOM中删除元素。作为一名Web设计师，我倾向于通过CSS隐藏这些元素，不过你也可以使用.remove()方法将这些元素从页面中彻底删除。

这个方法的使用非常简单，只需要在目标元素上调用.remove()方法。当DOM加载完成之后，这个元素就会被删除。我们也可以为.remove()方法指定一个选择器（后代选择器）参数，

这样就只有目标元素内匹配该选择器的后代元素被删除。

下面这个例子展示了`.remove()`方法的工作方式。参考图3-17可以看到该元素已经被从DOM中删除。图3-19展示的是`.content`元素被删除之后浏览器的输出。

```
<!doctype html>
<html>
  <head>
  <style>
    .google-icon {background:url(images/google-icon.gif)no-repeat;}
  </style>
  <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js'>
  </script>
  <script>
  $(document).ready(function() {
    $('.content').remove();
  });
  </script>

  <body>
    <div class="content">
      <p>I run 4 times a week.</p>
      <p>I lift weights 3 times a week.</p>
    </div>
  </body>
</html>
```

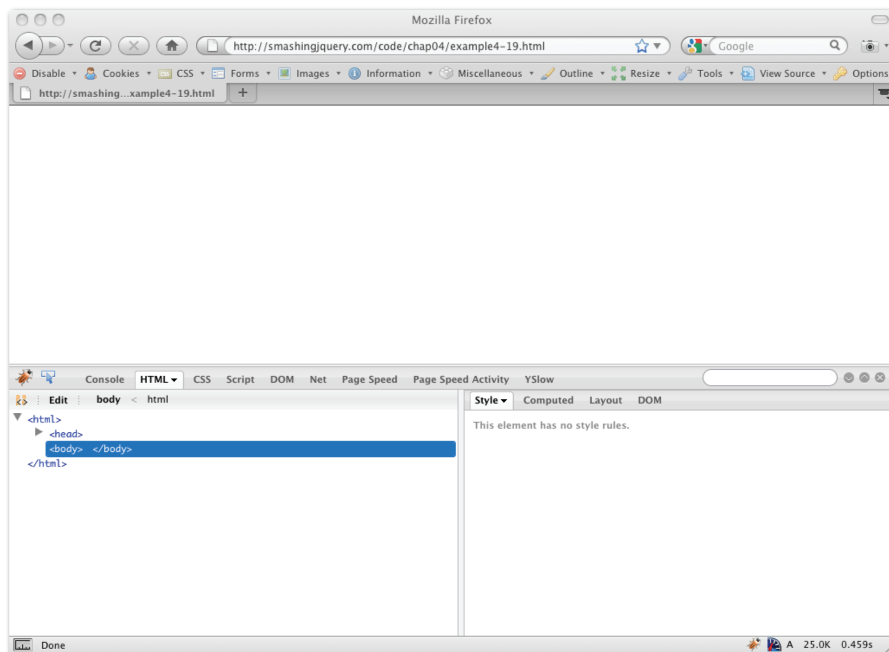


图3-19 `.content`类元素被移除之后浏览器的输出

6. 克隆HTML元素

有时候我们需要复制DOM中的某个元素，然后将它放到另一个地方。jQuery有一个.clone()方法，它负责克隆我们选中的任何元素。当一个元素被克隆之后，我们就可以使用.append()之类的方法将克隆的元素放置到任何地方，如下例所示：

```
$('.product').clone().append('.product');
```

3.3.5　在jQuery中使用CSS

jQuery仿佛总是与CSS协同工作。我常常会想，如果没了jQuery，我可该怎么活，特别是需要在脚本中密集使用CSS的时候。目前为止，我大都使用.css()方法为示例页面中的HTML元素添加CSS属性。使用.css()方法的缺点是它会在HTML中添加内嵌样式。如果你更喜欢干干净净的代码，我建议改用添加或删除样式类的方式。

表3-4概述了jQuery中所有可用的CSS方法。

表3-4　常用CSS方法及其功能

方　法　名	功　　能
.css()	获取或设定元素的CSS属性
.addClass()	添加CSS类
.hasClass()	检查元素是否拥有某个类
.removeClass()	删除CSS类
.toggleClass()	切换某个CSS类（没有就加上，有就删除）

1. 为DOM元素设置CSS属性

如果你是顺序阅读本章的话，就会对为DOM元素设定CSS样式很熟悉，因为我在每一个示例里都使用了.css()方法。这是一个非常有用的方法，特别是在调试布局时（为解决难缠的布局问题，常常不得不为某些元素加上边框进行调试）。

```
$('.container').css('border','1px solid #333');
```

2. 为DOM元素添加类

如果需要为某个DOM元素添加一个类，就用.addClass()方法。只要把类的名字作为参数传递给这个方法就可以了。不过记住，千万别给类的名字前面加上句点。

```
$('.container').addClass('active');
```

如果目标元素原来已经有一个类，则新类会被添加到该类之后。如果需要一次添加多个类，只需要用空格把这些类的名字隔开作为参数传入。

```
$('.container').addClass('active book');
```

看前面的例子，不要错以为只能为选中的类添加类，完全可以为任意选中元素（如ID和其他HTML标签）添加类：

```
$('#page-wrap').addClass('alternate');
```

3. 为DOM元素删除类

当然我们也有需要为元素删除类的时候。这时需使用方法.removeClass()，它和.addClass()的工作方式完全相同。

```
$('#page-wrap').removeClass('alternate');
```

如果没有为这个方法指定参数，则目标元素的所有类都被删除。

```
$('#page-wrap').removeClass();
```


只要将使用空格分开的多个类名字作为参数，就可以为目标元素一次删除多个类：

```
$('#page-wrap').removeClass('alternate main product');
```

4. 让DOM元素切换类

切换就是让某个东西开或者关。在开发动态Web应用时，.toggleClass()方法非常有用，在那些场合我们可能并不总是知道当时的开／关状态。

在下面的示例中，我为#page-wrap div元素切换.alternate类。如果div元素没有这个类，就把这个类加上，反之就把这个类去掉。

```
$('#page-wrap').toggleClass('alternate');
```

第 4 章 事 件

在jQuery中处理事件与前面提到的在HTML中处理CSS很相似，事件代码与目标元素互相分离。若使用原生JavaScript代码，所有的事件都直接嵌入到HTML代码中，这时维护自定义事件代码不但是一件麻烦事，而且很容易出问题。[①]虽然我们可以把所有功能函数存到一个外部文件当中，可是它们的调用代码却是以内嵌JavaScript的形式添加到元素标签之中。因此一旦修改了某个元素，就不得不更新有关的所有元素。使用jQuery，选定元素和事件处理代码与DOM彻底分离，存放于外部的JavaScript文件当中。

jQuery支持所有的原生JavaScript事件，但它让事件处理与Web应用的整合更加容易。在本章中，我们将一起使用jQuery API研究这些事件，弄清它们如何工作，最后一起看几个真实案例。

4.1 理解 jQuery 事件

jQuery事件是任何Web应用或Web页面的基础组成部分。在因特网刚刚兴起的时候，它只是一些静态文本、图像和超链接。随着技术不断进步，我们开始看到越来越多的动态页面，它们由后台数据库驱动。

随着后端技术越来越先进，前端技术也在不断改进，以Adobe Flash和JavaScript为首的这些技术为页面添加了新的交互层。超链接发展为翻转和层次菜单，图片则演进为支持修改功能的图库，静态文本则进化为强大的动态搜索引擎（比如Kayak旅行搜索引擎，它提供了基于地图的视图，见图4-1）。jQuery让我们用更少的代码，非常容易理解的语法，轻松地应用所有的原生JavaScript事件捕获来自用户键盘和鼠标的交互。这正是jQuery的风格！

---

① 公平地说，使用原生JavaScript也可以做到JavaScript代码与HTML代码分离。

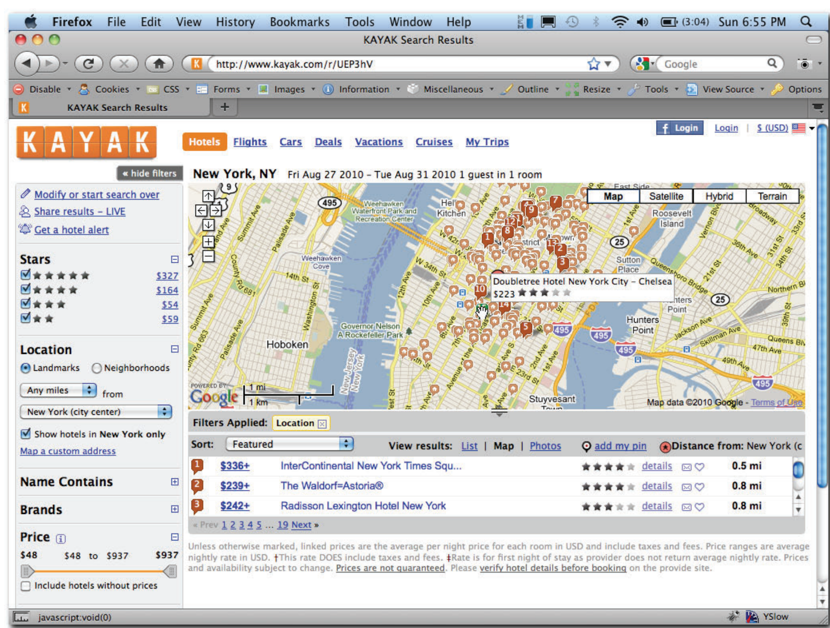


图4-1 在Kayak站点上，在事件的帮助下能够在地图上直观地预订酒店

4.2 使用文档和窗口事件

表4-1列出了jQuery常用的文档和窗口事件，这些事件相当重要，在jQuery编程中要牢记在心。表4-1简要列出了各个事件及其功能，既便于比较，又为我们在脚本中选择最合适的事件提供了参考。

表4-1 jQuery 文档和窗口事件

事件名字	事件功能
`ready()`	当HTML文档加载完成时触发
`load()`	HTML所有组件全部加载完成时触发
`unload()`	当浏览器窗口关闭或用户单击一个新链接或在地址栏输入新网址即将打开一个新页面时触发
`resize()`	当用户改变浏览器窗口大小时触发
`scroll()`	当用户滚动窗口时触发
`error()`	当HTTP请求遇到错误时触发

4.2.1 使用`.ready()`事件检测DOM是否完全加载

人们常用事件`.ready()`检查DOM是否加载完毕。当DOM准备好之后，这个事件将触发所有

包在`.ready()`事件处理函数中的JavaScript或jQuery代码。

为了确保只有DOM准备好之后才执行我们的jQuery代码，应该把这些代码放入`.ready()`事件处理函数，无论是HTML内嵌的方式还是外部文件的方式都没有问题。事件处理函数是这样一种函数，它和事件绑定在一起，当事件发生时就会被触发。在进行任何DOM操作之前，使用文档的`.ready()`事件处理函数确保DOM完全加载非常重要。如果忘记使用这个事件，代码很有可能不会像你期望的那样工作。

```
$(document).ready() {
  alert("The DOM is ready");
});
```

用这个事件来检测DOM完全加载再好不过，不过它并不等待页面的小部件（如图片或视频）成功加载。如果希望检查这些小部件是否加载完成，可以使用下面案例中的jQuery`.load()`事件。

4.2.2　使用`.load()`事件预加载图片

预加载图片相当实用。当页面上有很多图片时，如果我们采用了预加载技术，就可以做到当用户在页面上执行某些操作时这些图片已经就绪。在下面这个例子中，我演示了如何预加载5张图片，并在它们加载完成之后将它们插入到DOM当中。让用户等待页面中的图片一张张加载完毕是最糟糕的用户体验之一。下面这个例子有效地避免了这种情况。

我故意使用了较大的图片，它们的大小在100 KB到200 KB之间，并没有为Web应用做优化。

(1) 我使用原生JavaScript数组定义了一个名为`imgArray`的数组。这个数组保存着我打算添加到页面中的所有图片的文件名。使用这个数组容器，我能轻松地把它们添加到页面中，而不必写5条语句：

```
var imgArray =
["loc_portrait1.jpg",
 "loc_portrait2.jpg",
 "loc_portrait3.jpg",
 "loc_portrait4.jpg",
 "loc_portrait5.jpg"];
```

(2) 再定义两个变量用于下一步的`for`循环。变量`imgArrLength`保存着数组的长度，变量`i`用作`for`循环的循环变量。只要不是关联数组，数组的起始索引就总是0。

```
var imgArrLength = imageArray.length;
var i = 0;
```

(3) 构建一个`for`循环，循环条件为变量`i`小于`imgArrLength`。这个`for`循环将一直运行，直到`i`增长到数组的长度5为止。

```
for (i=0;i< imgArrLength ;i++) {
}
```

一个数组的索引总是从0开始，因此尽管这个数组一共有5个元素，数组的最大索引值却是4。下面的代码展示了如何从0到4设置数组索引：

```
imgArray[0] = "loc_portrait1.jpg";
imgArray[1] = "loc_portrait2.jpg";
imgArray[2] = "loc_portrait3.jpg";
imgArray[3] = "loc_portrait4.jpg";
imgArray[4] = "loc_portrait5.jpg";
```

(4) 在循环过程中，每次循环添加一个图片元素，使用.attr()方法把图片的src属性设置为图片文件名，把图片的id设置为img加上索引编号（如id="img1"），直到循环结束。

```
for (i=0;i <= imgArrLength;i++) {
   $('<img />').attr({'src':'images/'+imgArray[i],
'id':'img'+[i]}).load(function(){
   });
}
```


(5) 每当一个图片加载完成，就把它添加到DOM的.gallery容器。图片只有被添加到DOM才可被看到。图片何时被添加到DOM取决于它的加载时间。注意，图4-2中的渲染完成的源代码中图片文件的顺序并非与加载顺序（imgArray中的文件名顺序）相同，而基本上是文件大小顺序。

```
for (i=0;i <= imgArrLength;i++) {
  $('<img />').attr({'src':'images/'+imgArray[i], 'id':'img'+[i]}).load(function(){
    $('.gallery').append($(this));
  });
}
```

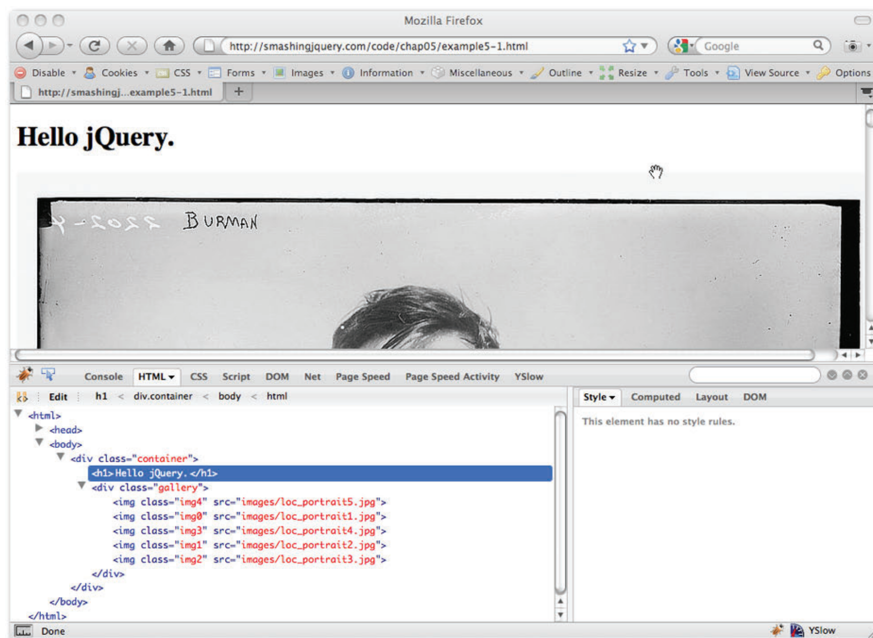


图4-2　jQuery成功预加载图片后，Firebug中的最终源代码输出

在前面这个例子中，注意this关键字的用法，在这里它是正在执行的方法中当前处理对象的引用。this关键字在这里引用的是.special-offer对应的元素对象，在下面这个例子中，this关键字引用的是当前正被处理的<img>标签对应的元素对象。this关键字极为有用，它在不同的场合有不同的含义。[1]图4-3演示的是this关键字如何工作。

```
$('.special-offer').bind('click' , function(){
        $(this).addClass('active');
});
```

图4-3　this关键字如何工作

(6) 把上面这些代码片段整合起来，放到你的IDE（Integrated Development Environment，集成开发环境）当中，就得到以下代码：

```
<!doctype html>
<html>
  <head>
  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js">
    </script>
  <script>
  var imgArray =
  ["loc_portrait1.jpg",
  "loc_portrait2.jpg",
  "loc_portrait3.jpg",
  "loc_portrait4.jpg",
  "loc_portrait5.jpg"];
  var imgArrLength = imageArray.length;
  for (i=0;i<= imageArray.length;i++) {
      $('<img />').attr('src', 'images/'+imageArray[i]).load(function(){
        $('.gallery').append($(this));
    });
  }
  </script>
  <body>
    <h1>Hello jQuery.</h1>
    <div class="gallery"></div>
  </body>
</html>
```

4.2.3　在用户离开页面时显示一条提示消息

你是否曾经希望在用户离开页面时显示一些内容？你是否希望再给他们一次机会，通过提供一些难以拒绝的东西吸引他们留下？jQuery的unload事件适合做这样的事。当用户单击浏览器的关闭按钮或者在地址栏里输入新网址即将打开新页面的时候，unload事件就会发生。你可能遇到过这样的网站，在你打算离开时，它就弹出一个又一个窗口，不让你关闭页面，是的，他们在

① 译者曾认真整理过this的应用场景，并据此写了一篇博文“JavaScript中七种函数调用方式及对应this的含义”，网址是http://t.cn/hDpAfn，欢迎有兴趣的读者阅读。

滥用unload事件。对一个想离开的用户来说，这可不是什么正确的做法。

不过，试想这样一个场景：一个人正在网站上填写一张表单，刚刚填写到一半，他突然打算放弃。这是使用unload事件的最好机会：我们可以显示一条提示信息，问问他是否真的打算离开。图4-4展示的是当用户打算离开时显示的一条提示信息。

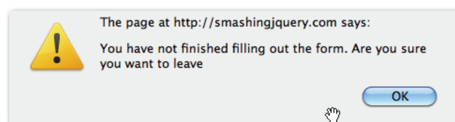


图4-4　在用户打算离开页面时显示一条提示信息

4

unload事件在一些老式浏览器中不能正常工作。在任何代码上线之前，我们都应该进行浏览器测试，确保这一事件在计划支持的所有浏览器中都能正常工作。Safari 3之前的版本以及很多Firefox 3的版本都有问题——在关闭页面时有可能不会触发unload事件，因此可能需要为这些较老的浏览器提供一种替代方案。

使用bind方法绑定unload事件处理函数。

```
$(window).bind('unload', function() {
  alert('You have not finished filling out the form. Are you sure you want to leave?');
});
$(window).bind('unload', function() {
  alert('You have not finished filling out the form. Are you sure you want to leave');
});
```

unload事件绑定代码可以简写，代码示例如下：

```
$(window).unload(function() {
  alert('You have not finished filling out the form. Are you sure you want to leave?');
});
$(window).unload(function() {
  alert('You have not finished filling out the form. Are you sure you want to leave');
});
```

4.2.4　使用error事件显示备用图片

我同时为几个电子商务站点工作，每个站点约销售200种产品，考虑到每种产品都需要3个不同尺寸的图片（缩略图、小图、大图），每个站点需要约600张产品图片。当某个产品页某个图片出现死链接，除非正好浏览到这个页面，否则我无法知道这个页面有图片丢失。利用jQuery的error事件，我就能检测到图片丢失并为它设置一张默认图片。

(1) 首先我们需要添加一个image标签，如下面的代码般简单即可。记着给这个标签一个独一无二的ID，以方便选中它。

```
<img src="images/loc_portrait10.jpg" id="portrait" />
```

(2) 接下来，使用#portrait选择器选中它。

```
$('#portrait')
```

(3) 为上一步选中的图片添加error事件处理函数。当浏览器无法正确加载文件时会触发error事件。

```
$('#portrait').error(funtion() {
});
```

(4) 最后，在事件处理函数中添加处理代码。在这个案例中，我们给匹配图片元素设置一张默认图片的src。

```
$('#portrait').error(funtion () {
  $(this).attr('src', 'images/default.jpg');
});
```

4.3　事件代理（委托）入门

在jQuery中，事件代理是指：把事件绑定到父级元素，然后等待事件通过DOM冒泡到该元素时再执行。图4-5演示了JavaScript事件的冒泡过程。如果你曾经使用过ActionScript 3.0，就会很熟悉事件冒泡的概念，因为在jQuery和JavaScript中使用的是类似的事件处理方式。在事件侦听过程中有两种触发事件的方式：事件捕获和事件冒泡。

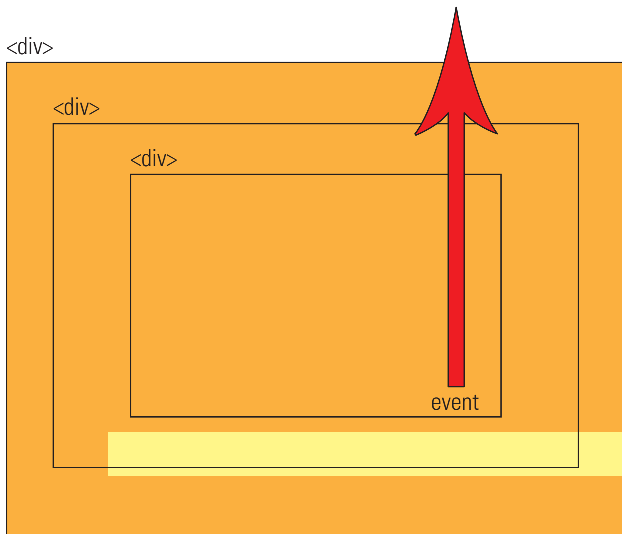


图4-5　事件冒泡演示

事件捕获过程是指事件在DOM中向后代元素下沉。事件冒泡则是指事件从发生事件的源元素通过DOM向上冒泡。除focus和blur事件之外的绝大多数事件都会冒泡。事件捕获和事件冒泡的不同在于传递方向相反，事实上事件冒泡更快，效率更高。jQuery使用事件冒泡的方式处理所有事件。

事件处理函数里面存放着事件发生时我们打算执行的代码。我们使用某个jQuery方法把事件

处理函数绑定到事件。自jQuery 1.4.2起，jQuery库提供了3个方法来绑定元素的事件处理函数，它们分别是bind()、live()和delegate()。

4.3.1　使用.bind()绑定事件处理函数

绑定事件处理函数的最基本的方式是使用.bind()方法。具体的用法是使用选择器选中一些元素，然后对结果集应用.bind()方法，并提供必需的两个参数，第一个是事件类型，第二个是事件处理函数。

```
.bind(event type, event handler)
```

在下面这个例子中，我们使用.bind()方法为所有.mylink元素绑定了click事件处理函数。当click事件发生时，会自动调用alertMe函数：

```
$(document).ready(function(){
  $('.mylink').bind('click', alertMe);

function alertMe() {
  alert("Hello World");
}
});
```

事件处理函数也可以是匿名函数，而且这正是我绑定事件处理函数的主要方式。下面的代码做的事情和上面的例子完全相同，不过它使用的是匿名函数，而非事先定义好一个具名函数。匿名函数，又叫*函数字面量*，特别适用于一段代码只在某具体上下文环境执行的场合。如果多个事件都需要执行同一段代码，则最好使用具名函数。

```
$(document).ready(function(){
 $('.mylink').bind('click', function(){
  alert("Hello World");
 });
});
```

.bind()方法仅适用于为DOM中已经存在的元素绑定事件处理函数。设想这样的场景：在DOM中有一个.box元素，你希望某个链接在每次被单击时复制这个元素，并把它添加到DOM中。我们可以为这个链接绑定一个适当的click事件处理函数。每单击这个链接一次，.clone()方法就会调用一次，复制box元素并将它追加到.container容器中。下面的代码演示了当DOM准备好之后，如何使用click事件在DOM中改变或追加元素：

```
$(document).ready(function(){
 $('.box').bind('click', function(){
   $(this).clone().appendTo('.container);
 });
});
<div class="container">
  <div class="box"></div>
</div>
```

> 受选择器的工作方式影响，你可能以为click事件也能作用于这些刚刚被添加到DOM中的新元素。错了！只有绑定事件那一刻DOM中存在的元素才会成功绑定click事件。click事件仅仅被绑定到了第一个元素上，因此我们只能不断地克隆第一个元素，而被克隆出来的这些元素却没有一个能够受click事件影响。

4.3.2 使用.live()绑定事件处理函数

.live()方法提供了一种相当灵活的捕获事件的方式。它的用法与.bind()极为相似：也是先使用一个选择器选中目标元素，然后在其上应用.live()方法。它同样需要两个参数，一个是事件类型，一个是事件处理函数。唯一的不同点在于.live()方法不仅作用于DOM中当前存在的元素，还作用于将来可能存在（动态生成）的元素。

```
.live(event type, event handler)
```

我们对上面的例子稍做修改，改用.live()方法绑定click事件到所有的元素，现有的及后续克隆出来的元素，都将支持单击事件。

```
$(document).ready(function(){
 $('.box').live('click', function(){
   $(this).clone().appendTo('.container');
 });
});
<div class="container">
  <div class="box"></div>
</div>
```

对那些需要经常在DOM中动态添加或删除元素的Web应用来说，.live()方法实在是好用极了。这就是DOM编程。

用.live()方法绑定事件的缺点是，它不支持下面代码示例所示的链接调用方式。这个代码无法如你期望的那样工作！[①]

```
$(document).ready(function(){
  $('.container').first().live('click',function() {
    $(this).clone().addClass('square').appendTo('.container:first');
  });
});

<div class="container">
  <div class="box"></div>
</div>
<div class="container">
  <div class="box"></div>
</div>
<div class="container">
  <div class="box"></div>
</div>
```

4.3.3 使用.delegate()绑定事件处理函数

要是能像.live()方法那样绑定事件，又支持链式调用其他方法，那该多好啊！.delegate()方法即是最佳解决方案，它非常灵活。它需要3个参数，第一个是选择器，第二个是事件类型，

① jQuery 1.7启用了新的.on()方法来绑定事件，完美地解决了这个问题。http://t.lava.cn/w/17823 上有个简单的介绍。

第三个是事件处理函数。它会将事件处理函数绑定到选择器参数匹配的元素上。传递给.delegate()方法的选择器参数用来找出具体的目标元素（或目标元素集合）。

```
.delegate(selector, event type, event handler)
```

.delegate()方法和.bind()及.live()的用法稍有不同，具体如下：

(1) 使用选择器选中目标元素，在本例中，目标元素是一个容器：

```
$('.container')
```

(2) 接下来用.delegate()方法绑定事件。我在.container元素中找出相册封面元素.box。然后，每单击一次，事件处理程序就克隆当前的box元素并将它追加到.container元素中。

```
$(document).ready(function(){
  $('.container').delegate('.box','click',function() {
    $(this).clone().appendTo('.container');
  });
});
```

下面这个示例演示了.delegate()如何支持链式调用。

本示例与前面那个示例的不同之处在于，我先使用.first()方法匹配了.genre元素，然后才调用.delegate()。结果是只有第一个.genre元素中的album-cover元素可以单击，并在被单击时将自身克隆并追加到.genre:first容器。

```
$(document).ready(function(){
  $('.genre').first().delegate('.album-cover','click',function() {
    $(this).clone().addClass('rock').appendTo('.genre:first');
  });
});

<div class="genre">
  <div class="album-cover"></div>
</div>
<div class="genre">
  <div class="album-cover"></div>
</div>
<div class="genre">
  <div class="album-cover"></div>
</div
```

4.4 捕获鼠标事件

当我们准备好在站点中添加用户交互功能时，会发现jQuery提供了各种鼠标事件，它们容易学习，而且可以轻易通过种种手段扩展，足以推动Web网站或应用的水平并使之更上一个台阶。

> 虽然jQuery没有提供原生的右键单击事件以提供上下文菜单，但第三方扩展实现了这一功能。

.bind()方法是jQuery中绑定事件处理函数的最常用方法，我会在接下来的示例中使用它。

表4-2是鼠标事件及其功能的一览表，表格之后是应用这些事件的一些实际案例，其中大多数案例体现了前面各章介绍的一些基本概念。

表4-2　jQuery中可以用`.bind()`方法绑定的鼠标驱动事件。

事　件　名	事件功能
`Click`	单击鼠标并释放将触发此事件
`dblclick`	双击鼠标触发此事件
`mousedown`	鼠标被按下那一刻触发此事件
`mouseup`	（鼠标按下后）松开那一刻触发此事件
`mouseenter`	鼠标进入某一元素区域时触发此事件
`mouseleave`	鼠标离开某一元素区域时触发此事件
`mousemove`	鼠标在某一元素区域内移动时触发此事件
`mouseout`	鼠标离开某一元素及其父元素时触发此事件
`mouseover`	鼠标进入某一元素及其父元素时触发此事件

4.4.1　通过单击鼠标触发添加或删除页面内容的行为

学习如何检测鼠标单击是jQuery中非常重要也非常热门的一个基本事件。因特网上链接和文本是很重要的内容，用户自然期望在你的站点上能到处单击。能够通过编程动态添加或删除页面内容非常有用，尤其是需要创建Web应用时。

如果使用原生JavaScript创建一个`click`事件，我们需要直接在元素上应用`onClick`动作。这里的`onClick`即事件处理函数，它在发生鼠标单击动作时调用一个函数。这种技术的缺点在于我们不得不把事件处理函数添加到具体的元素上，并到处嵌入JavaScript行为代码，这真是一团糟！。而jQuery则只需要选中打算添加`click`事件的元素。嵌入JavaScript会导致非语义的代码，就像下面这个例子一样（坏习惯）：

```
<a href="product.html" onClick="myFunction();">Paper Towels</a>
```

jQuery则是先选中打算添加`click`事件的元素，然后调用`.bind()`方法绑定事件。jQuery中绑定`click`事件处理函数的语法更简洁，并且不必在每个单独的元素标签中嵌入代码。

```
$(selector).bind('click', function() {
// 元素被单击后要执行的代码
});
```

以下解决方案阐明了如何通过单击鼠标添加并删除一个段落。

(1) 首先，创建一个`.content-link`链接和一个包含一些内容的`div`元素，在加载页面时这些内容默认是隐藏的。

```
<style>
.content {display:none;}
</style>
<h1>Hello jQuery.</h1>
```

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec gravida rhoncus
commodo. Aenean sit amet augue iaculis sem consectetur accumsan vitae a arcu. Quisque
quam diam, sollicitudin vel porta eget, vestibulum id justo. Vestibulum mattis metus
sed lorem adipiscing facilisis. </p>
<a href="#" class="content-link">Show More</a>

<div id="content">
  <p>Donec diam nisi, auctor sed tristique eget, pellentesque nec nisi. Sed vel libero
ipsum. Quisque semper, lectus in pulvinar tristique, nibh urna scelerisque augue, at
varius justo metus in orci. Suspendisse pretium arcu nec enim lacinia id cursus diam
rhoncus. Praesent pulvinar volutpat luctus. Vivamus cursus adipiscing tellus, id
fermentum turpis egestas lacinia. Aliquam tristique porttitor quam at pretium.</p>
</div>
```

(2) 选中.content-link链接，然后为其绑定click事件处理函数0。click事件绑定在这个链接上，当click事件发生时，就会自动执行下面代码中大括号括起来的部分。

```
$('.content-link').bind('click', function() {
  // click事件发生，就会执行这里的代码
});
```

(3) 要在该链接被单击后显示原来隐藏的内容，我们需要先选中这个div #content，然后调用.show()方法。

如果目标元素不可见，.show()方法就将它显示出来（把它display属性的值修改为block）；如果目标元素原来就是可见的，.show()方法就什么也不干。

```
$('#content').show();
```

(4) 如果click事件发生时希望隐藏内容，只需在目标元素上用.hide()方法调用代替.show()方法调用。

```
$('#content').hide();
```

(5) 还有一个替代方法，如果我们希望在click事件发生时切换目标元素的状态（显示/隐藏），就使用.toggle()方法。.toggle()方法会自动判断目标元素的当前显示状态，如果可见就将其隐藏，如果隐藏就将其显示出来。

```
$('#content').toggle();
```

(6) 我们将以上所有代码组织起来放到页面中，就得到以下代码。图4-6展示了该链接被单击之后的显示效果。图中Firebug插件处于激活状态，我们可以清晰地看到内容元素的display属性变成了block。

```
<!doctype html>
<html>
  <head>
  <style>
  body {font-family:arial;}
  #content {display:none;}
  </style>
  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js">
  < /script>
  <script>
      $(document).ready(function () {
```

```
            $('.content-link').click(function () {
                $('#content').toggle();
            });
        });
    </script>
    <body>
        <h1>Hello jQuery.</h1>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec gravida rhoncus
commodo. Aenean sit amet augue iaculis sem consectetur accumsan vitae a arcu. Quisque
quam diam, sollicitudin vel porta eget, vestibulum id justo. Vestibulum mattis metus
sed lorem adipiscing facilisis. Vestibulum sed ipsum ut nibh rutrum ultricies. </p>
        <a href="#" class="content-link">Show More</a>
        <div id="content">
        <p>Donec diam nisi, auctor sed tristique eget, pellentesque nec nisi. Sed vel
libero ipsum. Quisque semper, lectus in pulvinar tristique, nibh urna scelerisque augue,
at varius justo metus in orci. Suspendisse pretium arcu nec enim lacinia id cursus diam
rhoncus. Praesent pulvinar volutpat luctus. Vivamus cursus adipiscing tellus, id
fermentum turpis egestas lacinia. Aliquam tristique porttitor quam at pretium. In
ornare aliquet iaculis. Phasellus sit amet leo id urna tincidunt fringilla. Nullam ut
fringilla magna. Class aptent taciti sociosqu ad litora torquent per conubia nostra,
per inceptos himenaeos. Nunc leo arcu, sagittis eget volutpat at, bibendum at elit.
Vestibulum a nisi nec felis malesuada fringilla. Quisque vitae mauris nec sapien porta
pharetra eu vitae sapien. Cras bibendum eleifend malesuada.</p>
        </div>
    </body>
</html>
```

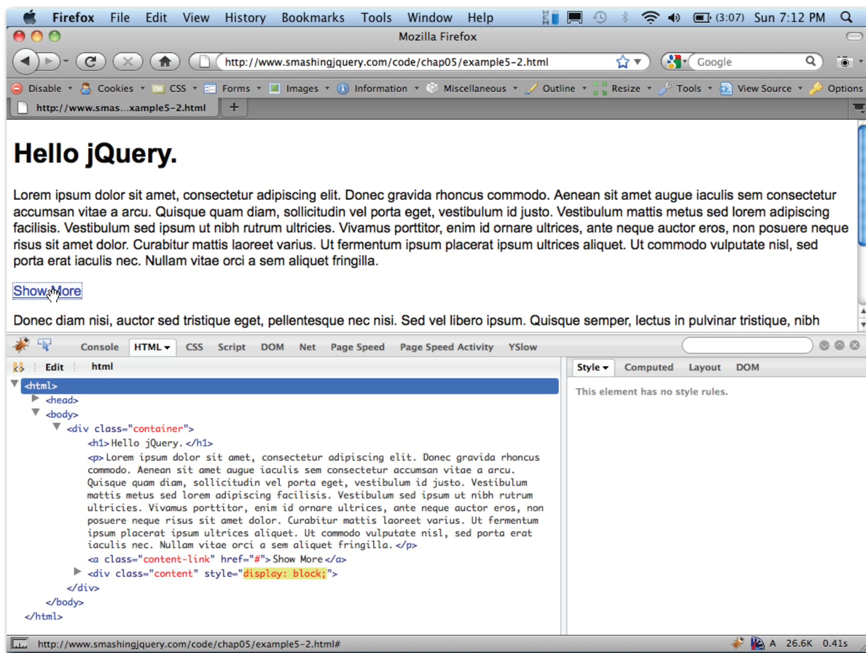


图4-6　单击链接后的屏幕显示，其中Firebug处于激活状态，内容元素的display属性已被.toggle()方法改变

4.4.2　双击事件

双击事件（dblclick）和click事件的工作方式相同，只有一点例外——必须双击鼠标才会触发这个事件。我并不经常使用dblclick事件，不过如果需要让Web应用运行起来像桌面程序，这个事件可就有大用了。

jQuery的dblclick事件与JavaScript原生dblclick事件一样，只是原生dblclick事件就像原生onclick事件一样需要直接添加到元素上。[①]

桌面软件常常要求双击鼠标以激活程序中的某个功能。在因特网上，鼠标双击可用于把产品添加到购物车。如果你的用户都是些因特网新手（他们习惯于双击以打开某个东西），也可以使用dblclick。不过，dblclick事件没有click事件那么常用。

4.4.3　利用鼠标悬停行为显示提示内容

在页面的链接、表单元素或一些其他元素上使用小提示或小泡泡状的信息窗口，能有效地帮助新用户访问站点，同时又可以使界面对老用户保持一致。例如，对那些不知道什么是信用卡上的CVV数字（信用卡核对码）的用户来说，提示信息可以是表单上的简单提示指令。另外，我们还可提供更进一步的交互说明（见图4-7）。图4-7展示了一个实际的由鼠标悬停行为触发的提示信息示例。

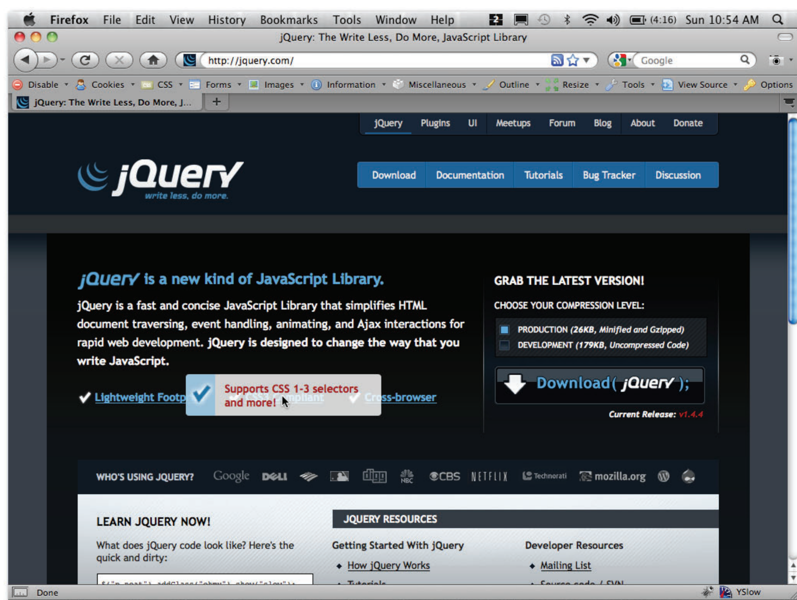


图4-7　jQuery官方站点上由鼠标悬停行为触发的提示信息窗口

① 作者再三强调这一点：原生事件只能直接 添加到元素标签内。请中文版的读者注意，这一观点是错误的，因为使用原生事件照样可以实现行为与结构的分离。

jQuery有一个悬停（hover）事件，它内部使用了原生JavaScript的mouseenter和mouseleave事件。hover事件不仅仅能用来显示提示信息，实际上它还可实现下拉菜单和图片预览（如图4-8所示Zappos站点上的例子）等很多功能。

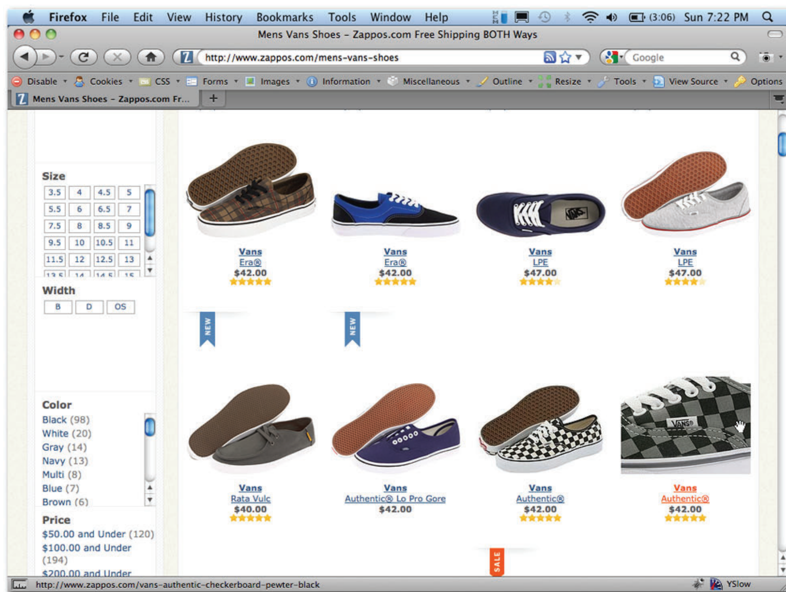


图4-8　Zappos站点，当鼠标悬停到一个产品之上时显示的预览图片

在jQuery中利用hover事件添加个性化的提示信息非常容易。如图4-9所示，在下面这个例子中，我希望做到当鼠标悬停在一张有说明的图片上时，可以将提示信息直接显示在图片下方。hover事件也可以应用于一个类似场合：不希望说明一直显示，只有用户将鼠标放到图片上时，才将说明显示出来。

(1) 为这张图片创建以下HTML，并给图片设置好title属性。之所以使用title属性，是为了确保即使用户禁用了JavaScript，一样可以看到图片的提示信息。这是一个非常好的渐进增强（参见第1章）的例子。

```
<h1>Hello jQuery.</h1>
<div class="cart">
  <h2>Shopping Cart</h2>
</div>
<div class="product">
<h3>Apple iPhone 4</h3>
<div class="product-image"><img src="images/iphone.jpg" title="Steve Jobs
  holds up the newest iPhone 4." alt="Steve Jobs"></div>
<p class="info">iPhone 4 is a GSM cell phone with a high-resolution display,
  FaceTime video calling, HD video recording, a 5-megapixel camera, and more.</p>
<p class="price">$299.99</p>
<div class="add-to-cart">Double-click to buy me</div>
</div>
```

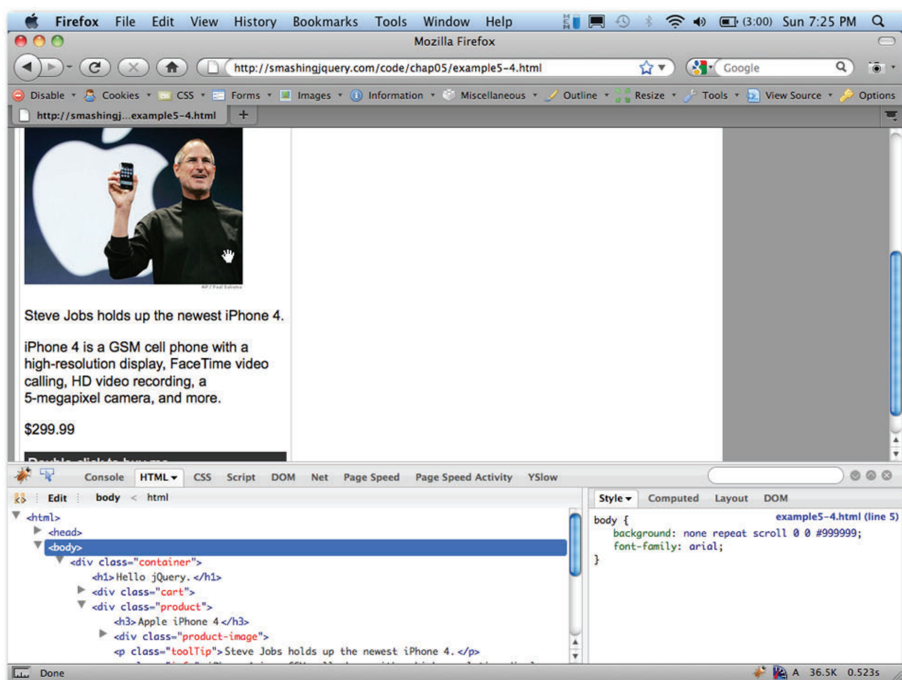


图4-9　鼠标在图片上悬停，信息提示随即显示；DOM改变时Firebug输出随之改变

(2) 下一步是添加document.ready事件处理函数，以确保在DOM准备好之后才执行我们的jQuery代码。

```
$(document).ready(function(){
});
```

(3) 选中.product-image元素内的图片元素，为它绑定mouseenter和mouseleave事件处理函数。这是一个后代元素选择器，因为我们最终匹配的是.product-image元素的后代元素。mouseenter和mouseleave事件确保用户移进和移出图片时会触发不同的行为。

```
$(document).ready(function(){
$('.product-image img').bind({
  mouseenter : function () { },
  mouseleave: function () { }
});
});
```

mouseover/mouseout与mouseenter/mouseleave的区别在于事件触发方式不同。

当鼠标进入/离开某个元素或它的后代元素时触发mouseover/mouseout事件。mouseenter/mouseleave事件当且仅当鼠标进入具体某个元素时触发，它不关心目标元素是否有子元素。mouseover/mouseout事件（由于事件冒泡）经常在不需要的时候不小心触发，从而导致一些脚本问题。

(4) 在本例中，目标元素是.product-image元素内的图片后代元素。当鼠标进入目标元素时，mouseenter事件触发。在这个事件处理函数中，我们创建一个名为tooltip的变量，使用这个变量保存利用.attr()方法获取的目标元素的title属性值。

```
$(document).ready(function(){
$('.product-image img').bind({
  mouseenter : function () {
    var toolTip = $(this).attr("title");
    $('.info').after('<p class="toolTip">'+toolTip+'</p>');
  },
  mouseleave: function () { }
});
});
```

(5) 第二条语句将tooltip变量的值用p.toolTip元素包裹，然后插入到DOM中div.info元素之后。

```
$(document).ready(function(){
$('.product-image img').bind({
  mouseenter : function () {
    var toolTip = $(this).attr("title");
    $('.info').after('<p class="toolTip">'+toolTip+'</p>');
  },
  mouseleave: function () { }
});
});
```

(6) 最后，当鼠标离开目标元素时触发mouseleave事件。这个事件中，我们在p.toolTip元素上应用.hide()方法将提示信息隐藏：[①]

```
$(document).ready(function(){
$('.product-image img').bind({
  mouseenter : function () {
    var toolTip = $(this).attr("title");
    $('.info').after('<p class="toolTip">'+toolTip+'</p>');
  },
  mouseleave: function () {
    $('p.toolTip').hide();
  }
});
});
```

在前面这个例子里，我使用了映射（键为事件类型，值为事件处理函数）参数来绑定mouseenter和mouseleave事件处理函数。使用这种方式绑定事件处理函数的优点是一次.bind()调用就能为同一元素绑定多个事件处理函数。

JavaScript对象字面量，又称为映射，是键值对的集合。它的语法是用一对大括号包着用逗号分隔的键值对，其中键和值分别用单引号括起来，键和值之间用冒号分隔。每个键/值对单独占

① 此处的处理是有问题的，会造成鼠标多次进入就产生多份p.toolTip，我们应该用.remove()删除p.toolTip，而不是隐藏它。

一行，就像下面这个例子一样：[①]

```
{
'name1' : 'value1',
'name2' : 'value2',
'name3' : 'value3'
}
```

有一种简便的方式使用mouseenter和mouseleave事件处理函数，效果完全相同。hover事件内部调用了mouseenter和mouseleave事件，却使用了更简洁的语法，且更容易使用，请看下面的代码：

```
$(document).ready(function(){
  $('.product-image img').hover(
    function () {
      var toolTip = $(this).attr("title");
      $('.info').after('<p class="toolTip">'+toolTip+'</p>');
    },
    function () {
      $('p.toolTip').hide();
    }
  });
});
```


4.4.4　利用mousedown和mouseup事件实现添加到购物车功能

拖放功能在网上越来越流行了。拖放允许用户在屏幕上以类似3D的风格拖动元素，把用户交互提高到了一个更高的层次。图4-10演示的是雅虎虚拟棒球运动界面灵活的拖放特性。

JavaScript原生API提供了onMousedown和onMouseup事件，它们可以嵌入任意元素以捕获这些事件。原生JavaScript事件与jQuery事件的主要不同之处在于，原生API都在事件名字前面添加了"on"这个单词。jQuery中的所有事件都未添加"on"，原生API中的onMousedown和onMouseup事件在jQuery中分别对应mousedown和mouseup。

在这一节，我将带你搞懂一些基本概念，然后一起构建一个简单的拖放系统。如果寻求一个健壮的拖放解决方案，jQuery UI提供了无数支持拖放的交互组件，我们可以轻易地将其整合到Web应用中。jQuery UI是一个基于jQuery的UI扩展库，提供了各种组件、界面元素以及主题，用于构建健壮的Web应用。如果想了解关于jQuery UI的更多信息，请访问jQuery UI的官方站点jqueryui.com。

在图4-11中，我复用了前面例子中的购物车代码。主要的不同在于之前使用dblclick事件将商品添加到购物车，这里则使用mousedown和mouseup事件模拟单击和释放一件商品。

---

① 此处描写并不十分准确。如果键是合法的JavaScript变量名，则可以省略引号。值如果是一个变量，就一定不能加引号。另外，每个键/值对单独写一行容易阅读，是好的编码习惯，但并非语法要求。多个键/值对完全可以写到一行之中。

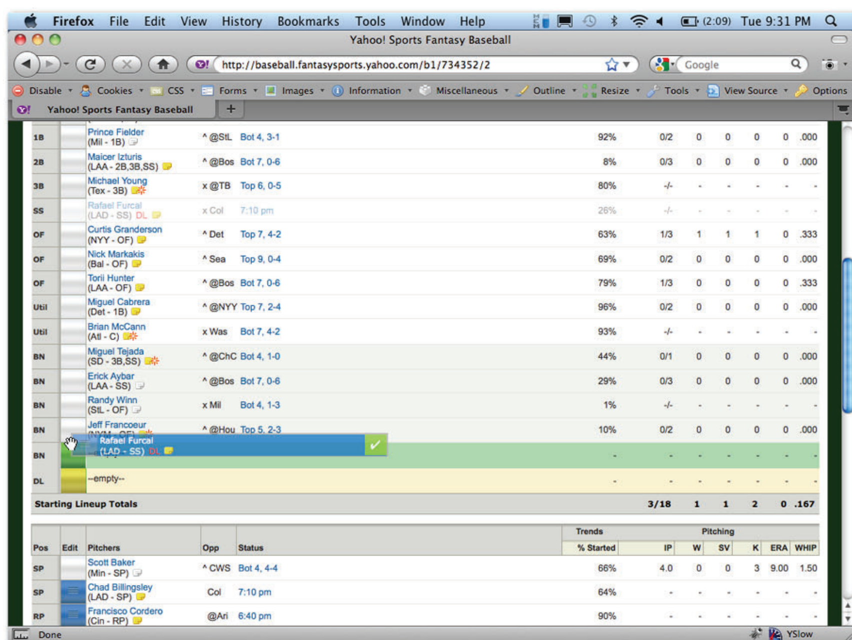


图4-10　雅虎虚拟棒球运动中的球队换人功能中使用了灵活的拖放界面

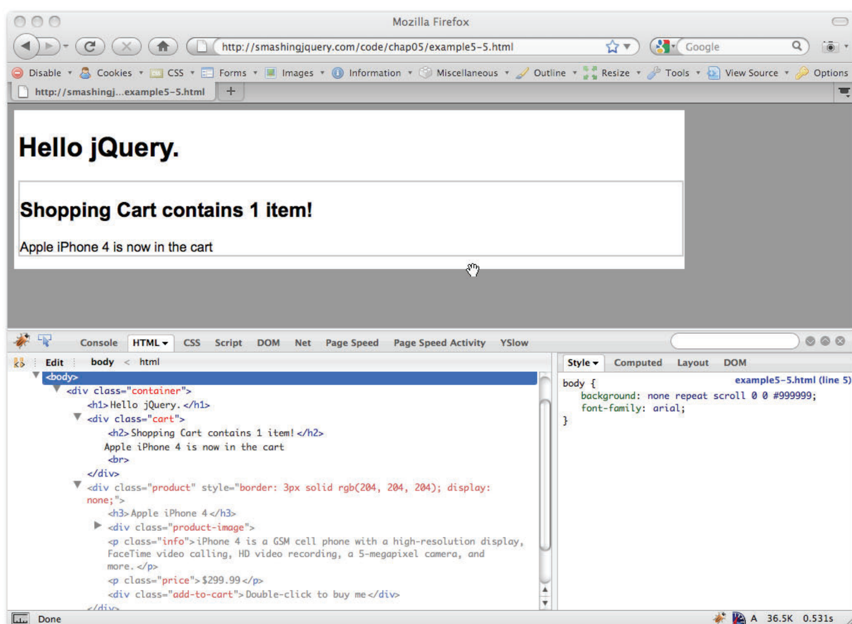


图4-11　Firebug激活状态下记录的简单拖放功能对DOM的改变（另见彩插图4-11）

(1) 首先在页面中创建必要的HTML元素。我虚构了一款产品和一个购物车，不过在本例中我们的脚本只与div.product打交道：

```
<div class="cart">
    <h2>Shopping Cart</h2>
  </div>
  <div class="product">
  <h3>Apple iPhone 4</h3>
  <div class="product-image"><img src="images/iphone.jpg" title="Steve Jobs
    holds up the newest iPhone 4."></div>
  <p class="info">iPhone 4 is a GSM cell phone with a high-resolution display,
    FaceTime video calling, HD video recording, a 5-megapixel camera, and more.</p>
  <p class="price">$299.99</p>
  <div class="add-to-cart">Double-click to buy me</div>
  </div>
```


(2) 接下来添加document.ready事件处理函数，以确保我们的代码在DOM准备好之后执行：

```
$(document).ready(function(){
});
```

(3) 下一步是添加mousedown和mouseup事件处理函数。我使用.bind()方法将事件处理函数绑定到.product元素：

```
$(document).ready(function(){
$('.product').bind({
  mousedown : function () { },
  mouseup: function () { }
});
});
```

(4) 当鼠标在元素上按下时，触发mousedown事件。这时我们允许元素被拖动，并将其边框改为3 px宽的红色实线边框：

```
$(document).ready(function(){
$('.product').bind({
  mousedown : function() {
    $(this).css('border','3px solid red');
  },
  mouseup: function() { }
});
});
```

(5) 在按下状态释放鼠标按钮会触发mouseup事件，这时在下面的代码中我们要做几处改动。首先，需要把边框恢复为3 px宽的浅灰色（#cccccc）实线边框，然后选中.cart元素，为其追加“Apple iPhone 4 is now in the cart”（苹果iPhone 4已放入购物车）文本，接着修改.card元素内的H2元素文本为“Shopping Cart contains 1 item!”（购物车中有一件商品），最后隐藏.product元素。

```
$(document).ready(function(){
$('.product').bind({
  mousedown : function() {
```

```
    $(this).css('border','3px solid red');
  },
  mouseup: function() {
    $(this).css('border','3px solid #ccc');
    $('.cart').append('Apple iPhone 4 is now in the cart<br>');
    $('.cart h2').text('Shopping Cart contains 1 item!');
    $(this).hide();
  }
});
});
```

关键之处在于我们必须能检测到鼠标按下和释放事件，并为这两个事件编写截然不同的事件处理函数。图4-11展示的是Firebug激活状态下记录下的简单拖放功能对DOM的改变。

所有鼠标事件都有相应的简写形式，可使开发者以更简洁的方式在Web站点或应用程序中使用jQuery功能。

下面是使用`.bind()`方法绑定`click`事件处理函数的普通写法：

```
$('.container').bind('click', function(){});
```

将上面这个事件绑定改用简写形式书写：

```
$('.container').click(function(){});
```

使用简写形式的缺点在于无法做到像下面这样一次为多个事件绑定处理函数：

```
$('.container').bind('click dblclick', function())};
```

4.4.5　实现图片翻转效果

在jQuery中实现翻转效果是一件相当享受的事。大约10年前我第一次使用HTML和CSS实现图片翻转效果，记得当时使用的是Marcomedia HomeSite编辑器，这是一款支持所见即所得的编辑器，它生成那些令人发疯的JavaScript代码（大约有25行，天知道为什么需要那么多！），只为了实现简单的图片翻转。在下面这个解决方案里，我将演示给你看：同样做这件事情，现在只需要4行代码！

(1) 首先需要为图片添加一个`class`，这样我们就能使用jQuery选择器方便地找到它：

```
<img src="images/rollover_off_btn.gif" class="button" alt=""/>
```

(2) 接下来设置`.hover()`事件：

```
$('.button').hover(
  function() {
  },
  function() {
});
```

(3) 使用`.hover`事件内的两个事件处理函数改变图片的`src`属性。第一个事件处理函数对应`mouseenter`状态，第二个对应`mouseleave`状态。如下所示，我们使用`.attr()`方法改变`src`属性的值（图片路径）：

```
$('.button').hover(
  function() {$(this).attr('src','images/rollover_on_btn.gif');},
  function() {$(this).attr('src','images/rollover_off_btn.gif');
});
```

下面是本解决方案实现图片翻转效果的全部代码：

```
<!doctype html>
<html>
  <head>
  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"> <
    /script>
  <script>
  $(document).ready(function(){
    $('.button').hover(
      function() {
        $(this).attr('src','images/rollover_on_btn.gif');
       },
      function() {
        $(this).attr('src','images/rollover_off_btn.gif');
    });
  });

  </script>
  <body>
      <img src="images/rollover_off_btn.gif" class="button" alt="Button"/>
  </body>
</html>
```

4.5　捕获表单事件

类似鼠标事件使用`.bind()`方法绑定事件处理函数，表单事件也一样可使用`.bind()`方法绑定事件处理函数。在Web站点和应用中表单极其常见，几乎我们使用的每一个Web应用都有表单，从基本的搜索到成熟的三步注册表单，表单几乎可应用于任意的场合。在一些应用场合，较长的表单要求用户填写很多的信息，这不但有可能把用户吓到，还会导致难以清理的数据。在触发一些动作时，比如当用户离开某个表单项时触发表单验证，或者提醒用户某个表单项需要特别注意时，jQuery事件能够帮上大忙。善用表单事件能以多种方式增强用户体验。

jQuery提供的少量几个表单事件能有效地增强用户填写表单的体验，这几个事件在原生JavaScript中也很常用。表4-3列出了这些事件的名字及其功能。

表4-3　jQuery表单事件名字与功能一览表

表单事件名	表单事件功能
`change()`	表单项的值改变时触发此事件
`focus()`	敲TAB键进入某个文本域或者选择某个文本域（得到焦点时）时触发此事件
`focusin()`	一个元素或它的子元素得到焦点时触发此事件
`focusout()`	一个元素或它的子元素失去焦点时触发此事件
`blur()`	文本域（input:text）或文本框（textarea）失去焦点时触发此事件

（续）

表单事件名	表单事件功能
select()	元素内的文本被选中时触发此事件
submit()	表单提交时触发此事件（或者单击了Submit按钮，或者按下了回车键）
reset()	表单使用input type="reset"重置表单数据时触发此事件

4.5.1 得到焦点时为表单元素添加边框

focus事件常常用来指导用户填写表单项。当一个元素获得焦点时触发focus事件，这种元素通常是一个表单元素。当用户开始输入数据时，我们可以利用focus事件的处理函数，做一些事情。

在下面这个例子中，我们演示了当用户单击表单的输入框时，利用focus()事件为该表单元素添加CSS边框的功能。如果用户需要依次填写很多表单项，这一功能非常有用。它会提醒用户目前正在填写的是哪个表单项。

(1) 第一步还是准备HTML。我们添加一个文本框并为它设置id属性的值——firstname。

```
<input type="text" name="firstname" id="firstname"/>
```

(2) 选中#firstname元素，然后为其绑定focus事件处理函数。

```
$('.firstname').bind('focus', function()};
```

(3) 在事件处理函数中为目标元素设置边框属性（添加边框）。

```
$('.firstname').bind('focus', function(){
  $(this).css('border','1px solid red');
})};
```

现在input#firstname已绑定focus事件处理函数，无论是通过鼠标单击还是敲TAB键进入这个文本域，它都会自动显示1 px的红颜色实线边框。瞧，就这么简单。

4.5.2 焦点离开输入框之后显示消息

blur事件与focus事件相反，它在一个文本框或元素失去焦点时触发。在真实的世界里，blur事件常用于验证用户输入的电子邮件地址是否合法，或者为文本框画上红色边框等。我将在第8章专门讲解如何创建交互性好的表单元素。

4.6 捕获键盘事件

除了事件来源是键盘这一点之外，键盘事件与鼠标事件非常相似。当我们按下一个键或者用键盘修改了文本域的内容时就会触发键盘事件。键盘事件常常用于浏览器游戏和表单验证。在那些必须注册才能使用的站点上，在我们输入用户名的过程中，用户名验证脚本可能已经在键盘事

件的驱动下开展工作了。表4-4列出了这些事件以及它们的功能。

表4-4　jQuery键盘事件一览表

事件名字	事件功能
keydown()	当一个键被按下时触发此事件
keypress()[①]	当一个键被按下一次或多次时触发此事件
keyup()	当一个键弹起时触发此事件

对任何Web表单来说，验证用户输入都至关重要。来自用户的输入数据需要验证无误才能确保（储存的）数据干净可用。利用表4-3中的change事件可以在文本框中的内容发生变化时执行检查任务。很多编程爱好者觉得仅有JavaScript验证不够安全。实际上是否需要增加服务器端验证，要看验证的是些什么数据。

图4-12展示了一个常见应用，它是Twitter的输入框，这个文本框至多允许输入140个字符。每输入一个字符，允许输入的字符数就减少一个。页面上有一个提示文本，告诉你还可以输入多少字符。

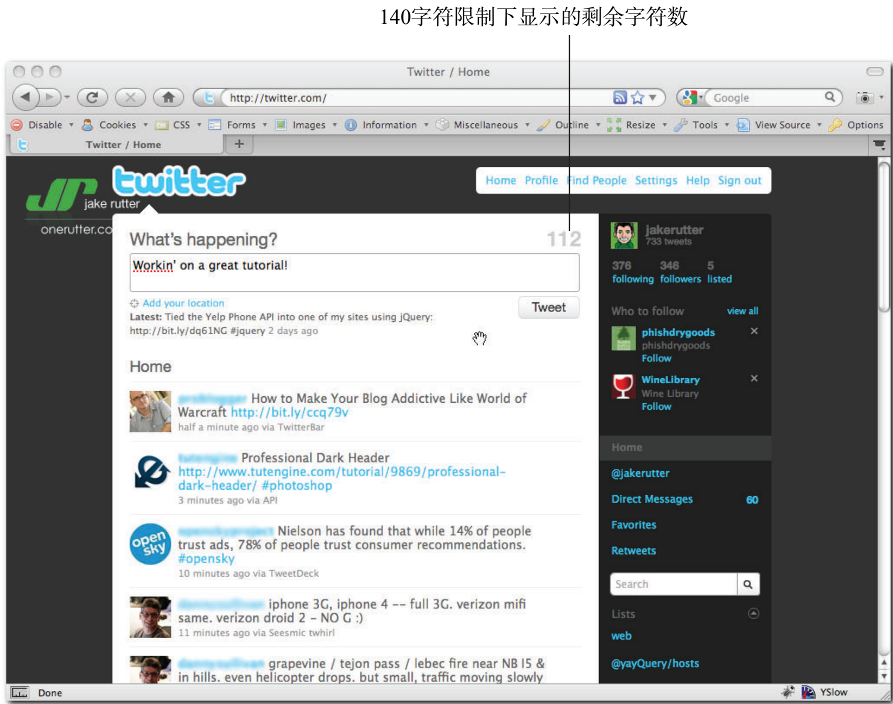


图4-12　Twitter状态输入框的字符数限制功能

① 关于keypress事件，详见译者的博文："keypress事件备忘"，地址为http://t.lava.cn/blog/21689。

和其他事件一样，我们使用.bind()方法绑定change事件：

```
$('.container').bind('change', function());
```

在下面这个例子中，我演示给大家如何编写类似Twitter上显示剩余字符提示的脚本。这个例子的最终显示效果见图4-13。

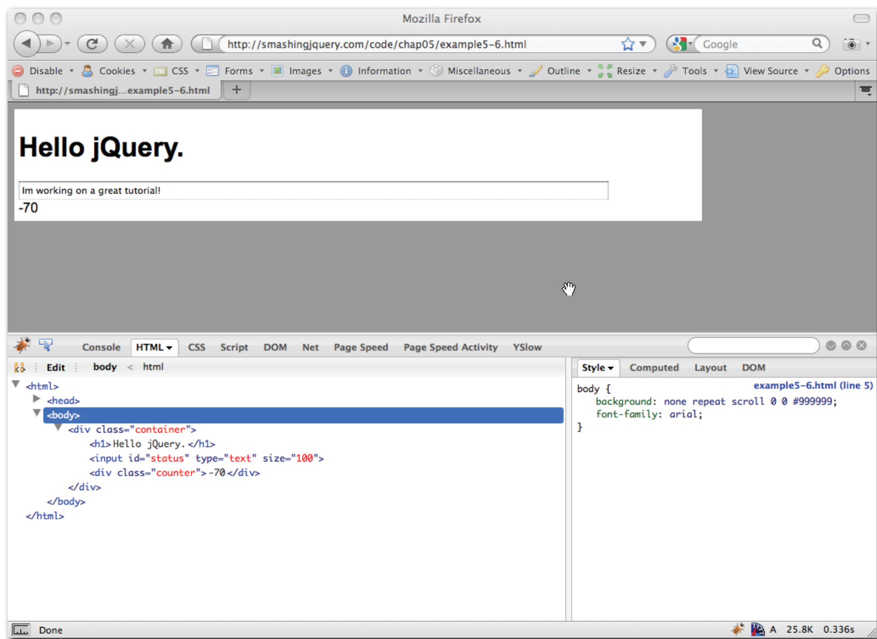


图4-13　输入字符到文本框的过程中脚本不断执行，计数器数字不断减小

(1) 第一步是准备HTML。我们使用一个textarea标签，并把它的id属性设置为status。我们会把change事件处理函数绑定到这个输入框，以检测用户输入的字符。

```
<textarea cols="50" rows="5" id="status"></textarea>
```

(2) 添加一个div空元素，设置它的class属性为counter。在输入框中输入数据时，剩余字符数将显示在这里。

```
<div class="counter"></div>
```

(3) 添加一个变量maxNum，它负责保存最多允许输入的字符数，我们把它的值设置为100。

```
var maxNum = 100;
```

(4) 选中#status元素，然后为其绑定keypress事件。当有键被按下时，就会触发keypress事件，因此 在表单输入元素得到焦点后，它特别适合用来执行检测输入。

```
$('#status').bind({
    keypress : function() {
});
```

(5) 当keypress事件发生时，我需要拿到#status输入框的值。我使用变量inputText保存这个值，还创建了一个名为numChar的变量保存inputText变量的长度（字符数），然后创建了一个名为charRemain的变量保存maxNum减去numChar的结果（剩余字符数）。

```
$('#status').bind({
  keypress : function() {
            var inputText = $(this).val();
            var numChar = inputText.length;
            var charRemain = numChar - maxNum;
  }
});
```

(6) 搞定这些变量之后，需要检查numChar是否小于等于maxNum，为此我添加了一个条件语句。如果这个比较表达式为真，我就选中.counter元素并修改它的文本为charRemain（剩余字符数）。

```
$('#status').bind({
  keypress : function() {
    var inputText = $(this).val();
    var numChar = inputText.length;
    var charRemain = numChar - maxNum;
    if (numChar <= maxNum) {
      $('.counter').text(charRemain);
    }
  }
});
```

(7) 最后一步，添加一个else if子句检查numChar是否大于maxNum。如果这个表达式为真，则使用event.preventDefault()方法阻止用户输入更多内容。event.preventDefault()方法等同于原生JavaScript语句return false，用来阻止事件的默认行为（字符上屏）。

```
$('#status').bind({
  keypress : function() {
            var inputText = $(this).val();
            var numChar = inputText.length;
            var charRemain = numChar -- maxNum;
            if (numChar <= maxNum) {
                   $('.counter').text(charRemain);
            }
    else if (numChar > maxNum){
      event.preventDefault();
    }
  }

});
```

4

第5章 用特效为Web站点添色

最近几年，JavaScript特效逐渐成熟，已经成为多年来一直在线上特效领域占据统治地位的Adobe Flash的有力竞争者。以前要在Web站点上使用图片相册、动画菜单或动画视频效果，非Flash不行，今天人们却越来越多地使用JavaScript，以便更容易地兼容多个浏览器和移动设备。JavaScript特效的日益流行已经成为促使Web设计师和开发者使用jQuery特效API的主要动力。

jQuery总能充分挖掘原生JavaScript效果以提供健壮的、易于整合到任何站点的功能，这一点对我们来说并不陌生。因为这些效果都由jQuery提供，所以极其容易使用，不可避免地成为广大Web设计师和开发者的一个热门选择。

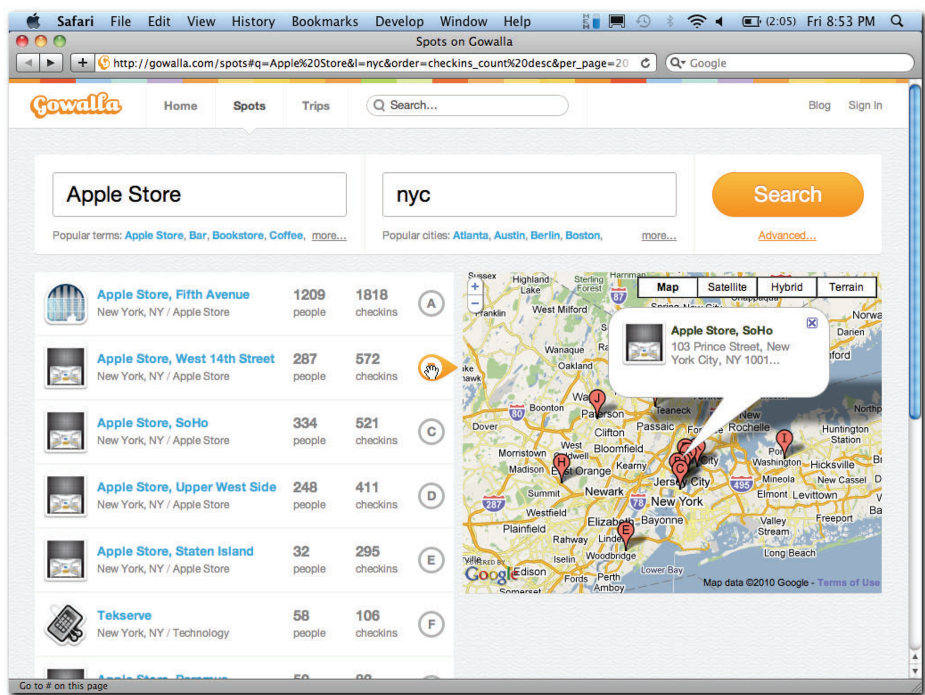
在本章中，我们将一起透过jQuery API研究这些特效，讲解每种特效的使用，然后一起研究几个真实的应用场景。

5.1 jQuery 特效能做些什么

作为一名Web设计师和前端开发人员，让站点的用户界面好用是我们的职责。在一个页面上我们经常需要让一些元素在屏幕上显示出来或者隐藏起来，以容纳更多内容。用户需要并且期待Web站点能即时展现：他们不喜欢笨重的站点，一个又一个页面永无休止地加载。

Facebook，极受欢迎的社交网络，拥有5亿多用户，它的界面由JavaScript驱动，不但有趣而且交互性极佳。登录Facebook，无需打开任何新页面你就可以和朋友们聊天，或者查看朋友们的新鲜事。这么好的用户体验是通过JavaScript特效（显示、隐藏或者动画）实现的。类似Facebook的诸多Web站点正在为用户期望的线上体验设立高标杆。

基于地理定位技术的应用程序开始赢得眼球，变得越来越流行。多数这样的站点前端都使用JavaScript整合了类似谷歌地图的技术。图5-1展示的是Gowalla站点，一个基于地理定位的社区。



© 2010，Gowalla公司

图5-1　Gowalla网站，一个基于地理定位的社区

jQuery提供了一些基本效果，如显示、隐藏、滑动和淡入淡出。表5-1简要列出了这些基本效果，这些函数的用法都很相似，使用相同的可选参数。

表5-1　jQuery基本效果

方　法　名	方法元素
`show()`	显示一个元素
`hide()`	隐藏一个元素
`toggle()`	依据当前状态切换显示／隐藏状态
`slideDown()`	以向下滑动展开的方式显示元素
`slideUp()`	以向上卷动的方式藏起元素
`slideToggle()`	以滑动／卷动方式切换元素的展开／收起状态
`fadeIn()`	元素以淡入的方式显示出来
`fadeOut()`	元素以淡出的方式消失
`fadeTo()`	淡入或淡出到某个透明度

5.2　使用.show()和.hide()方法显示或隐藏元素

使用jQuery显示/隐藏元素是最简单的特效。在前面几章的例子中我已经用过这两种特效，不过一般来说，这些特效经常用于click事件处理。它广泛应用于整个因特网。图5-2显示的是谷歌Gmail使用显示／隐藏特效显示一个层，展示其支持“拨打电话”这一新功能。

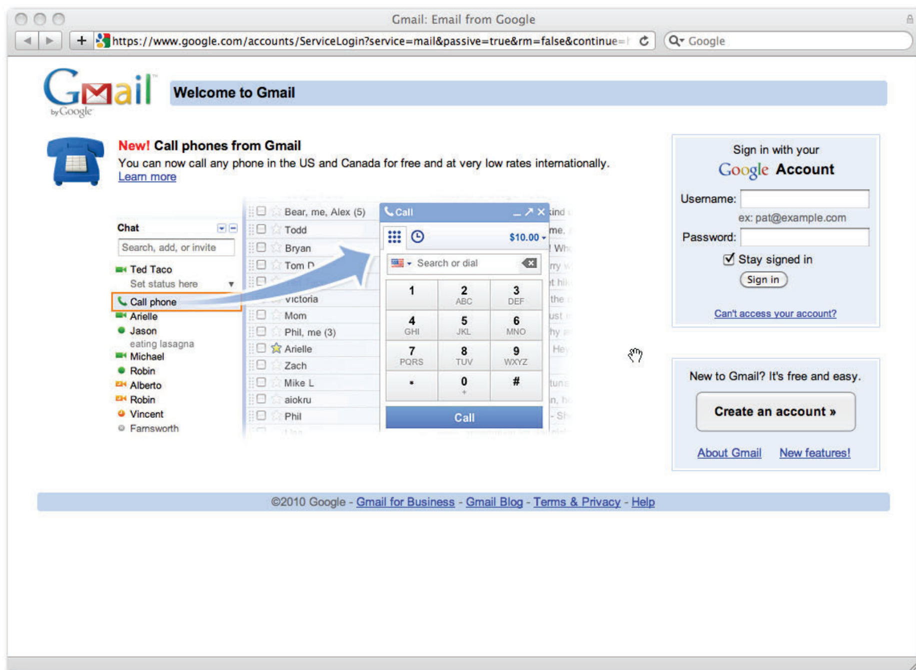


图5-2　Gmail使用显示/隐藏特效展示推广其“拨打电话”的新功能层

使用选择器选中元素，然后调用.show()或.hide()方法。这两个方法都支持两个可选参数。duration参数决定动画的持续时间，可取的值有fast、slow或者以毫秒为单位的整数（如600、200、700等）。callback参数是一个回调函数，它会在.show()执行完成时被调用。

```
$(selector).show(duration, callback)
```

下面这个例子为一个链接绑定了click事件处理函数。当单击这个链接时，div.recipe元素就显示出来。这是使用.show()方法的最基本形式。

```
<style>
.recipe {display:none;}
</style>
$('.recipe-name').bind('click', function() {
  $('.recipe').show();
});
```

```
<a href="#" class="recipe-name">Key Lime Pie</a>
<div class="recipe">Key lime pie is an American dessert made of key lime juice, egg
yolks, and sweetened condensed milk in a pie crust.</div>
```

jQuery `.show()`方法为div元素设置了`display:block`行内样式，从而使它显示出来。图5-3展示的是上述示例代码在浏览器中的渲染结果。

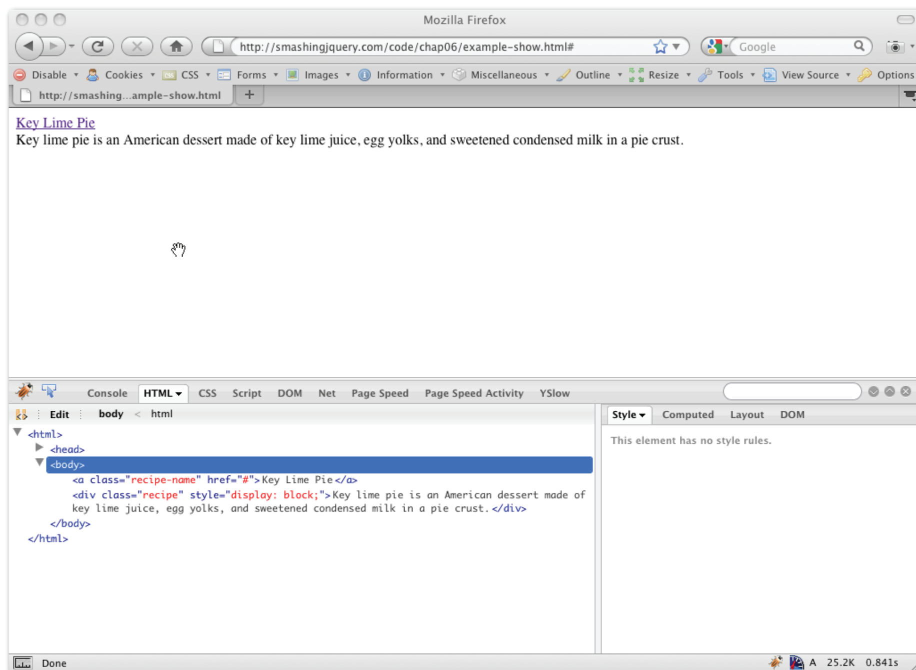


图5-3 上述示例代码在浏览器中的渲染结果

`.hide()`方法工作方式与`.show()`几乎完全一样，只有一点不同，它为选中元素添加行内样式`display:none`，从而让该元素隐藏而不是显示出来。

```
$('.recipe-name').bind('click', function() {
  $('.recipe').hide();
});

<a href="#" class="recipe-name">Key Lime Pie</a>
<div class="recipe">Key lime pie is an American dessert made of key lime juice, egg
  yolks, and sweetened condensed milk in a pie crust.</div>
```

如果希望微调show特效的显示速度，在调用`show()`方法时可传入`fast`/`slow`关键词，也可传入一个以毫秒为单位的整数。

```
$('.recipe').show('slow');
```

你也可以传入一个回调函数参数给`.show()`方法，当特效完成时会自动调用该函数。

5.2.1　结合.show()方法和cookie让一条消息在站点上只显示一次

设想这样一个情景：我们需要向用户显示一条专用的提示或者消息，且只希望让他们看到一次。我经常在Basecamp（一个在线项目管理工具）上看到类似的信息，它会在我登录时提示有一些新功能。我们可以使用.show()方法，同时在用户的机器上种一个cookie，利用这个cookie阻止同一用户在同一台计算机上再次看到这条消息。

首先，我们生成这条消息并"烘烤"这块cookie。

(1) 为我们想要显示的消息准备HTML，其中有一个链接，用户单击它就可以隐藏消息。页面上还有一个链接，当消息隐藏之后，用户可以通过单击它再次看到这条消息。

```
<a href="#" class="special-offer">View this special offer!</a>

<div id="message">
  Special Offer for Members! 50% off your first purchase.<br/>
  <a href="#" class="hide">Hide this message</a>
</div>
```

(2) 准备用来显示提示信息的click事件处理函数，它负责显示提示信息，并在显示完成后调用hideMessage函数（稍后我们会编写这个回调函数）。在该事件处理函数中，我们选中#message元素，并调用.show()方法。我们在这儿不需要duration参数，直接把回调函数hideMessage作为参数传递给.show()即可。

```
$('.special-offer').bind('click', function(){
  $('#message').show(hideMessage);
});
```

(3) 再准备一个用来隐藏信息的click事件，它负责隐藏这条消息，并在隐藏完成后调用hideMessage函数。

```
$('.hide').bind('click', function(){
    $('#message').hide(hideMessage);
  });
```

(4) 接下来，我们来编写hideMessage回调函数，在用户看完提示信息之后，它负责在用户计算机上种一个cookie。

```
function hideMessage() {
}
```

(5) 创建一个名为hideCookie的cookie，将它的过期时间设置为30天。我们在这儿使用了JavaScript中的date对象。这是一个很好的混合使用JavaScript原生代码与jQuery代码的例子。图5-4展示的是Firefox中设置完该cookie[①]之后的cookie信息截图。

```
function hideMessage() {
  var expirDate=new Date();
  expirDate.setDate(expirDate.getDate()+30);
  document.cookie = "name=hideCookie;expires="+expirDate.toUTCString();
}
```

① jQuery有一个cookie插件，利用这个插件我们可以更方便地读取和设置cookie。

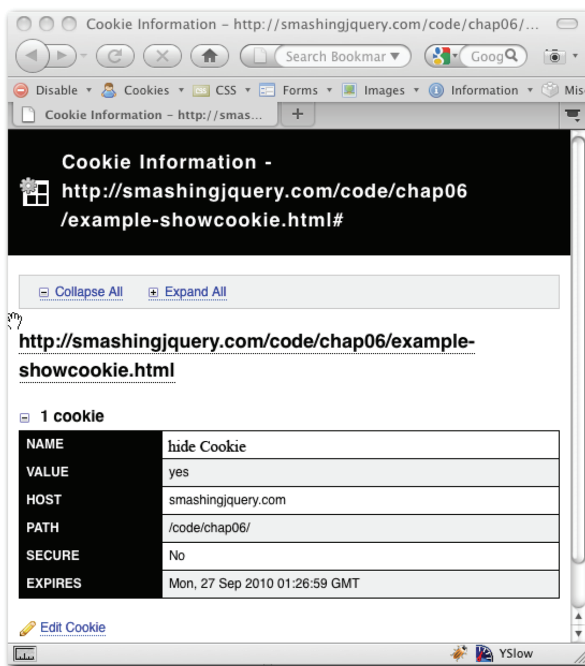


图5-4　显示在Firefox中的cookie和它的值

现在我们能够做到显示和隐藏信息，还能设置cookie。可是该如何响应那些已经看过这条信息的用户？如果他们第二天又来这个站点，他们一定不希望又一次看到这个信息。我们可以再写一个函数检测是否存在这个cookie，若cookie存在则隐藏这条信息。在页面`load`事件发生时，我们执行这个函数，正如图5-5所示。

(1) 声明一个变量保存页面cookie。利用JavaScript的cookie对象，使用`document.cookie`可得到一个cookie。

```
var messageCookie = document.cookie;
```

(2) 写一个条件语句检测`messageCookie`是否有值。若`messageCookie`有值则隐藏这条提示信息，否则什么也不做。[①]

```
if (messageCookie) {
   // 如果消息cookie存在，则隐藏提示信息
   $('.special-offer').hide();
 }
 else {
   // 什么也不做
 }
```

① 此处检测cookie的手段非常粗放，假设该站点并没有设定`hideCookie`这个cookie，但有其他用途的cookie，提示信息也会自动隐藏，这显然是我们不想要的。利用jQuery的cookie插件可以轻松拿到具体的cookie。

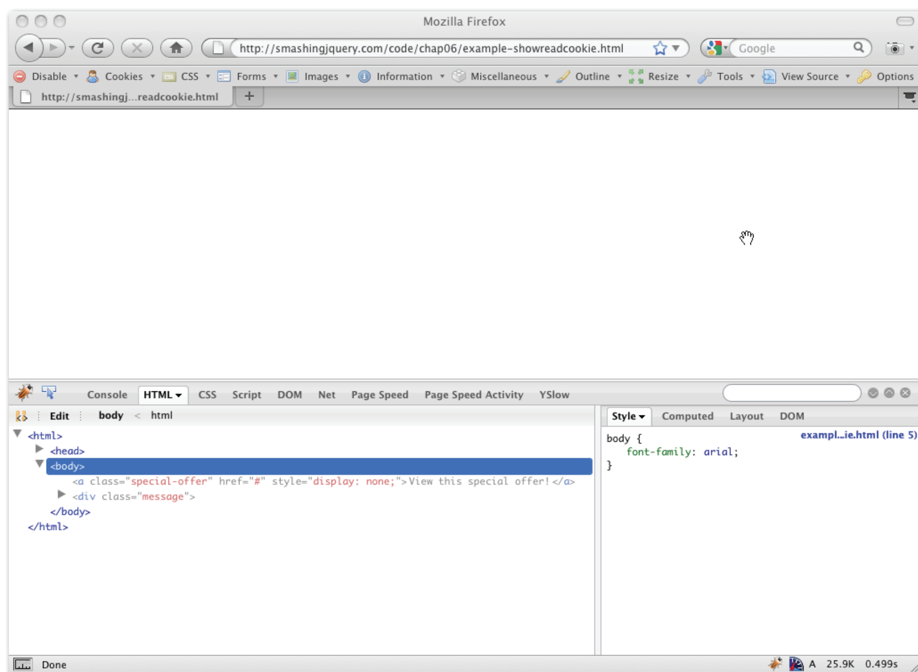


图5-5　单击链接后，链接和信息都看不到了

5.2.2　切换元素的显示状态（显示 / 隐藏）

有时我们需要切换一些内容的显示状态，jQuery提供了一个很棒的解决方案，即.toggle()。.toggle()方法为调用该方法的元素的click事件绑定了一个事件处理函数，让我们能够基于目标元素的当前显示状态控制目标元素的显示。下面代码片段中最重要的部分是利用CSS使.recipe元素隐藏。（display属性的值决定.toggle()方法如何工作。）

```
<style>
.recipe {display:none;}
</style>
$('.recipe-name').toggle(
 function() {
  $('.recipe').show();
 },
 function() {
  $('.recipe').hide();
 }
);
<a href="#" class="recipe-name">Key Lime Pie</a>
<div class="recipe">Key lime pie is an American dessert made of key lime juice, egg
  yolks, and sweetened condensed milk in a pie crust.</div>
```

5.3　滑动元素

在网上冲浪，经常会见到滑动效果，特别是在那些图库类应用中，我们经常看到图片滑进视野或滑出视野。此外，最近在Facebook和Twitter上流行的实时会话，在页面上新消息的滑入滑出更是司空见惯。图5-6是Twitter的首页（`twitter.com`），当发表新消息时，新消息就以滑动的方式显示出来。每发表一条消息，这条消息就从顶上滑动下来，将页面上原有的消息向下推，一条接一条，直到滑出视野范围。



图5-6　在发表新消息时，Twitter首页使用了滑动效果

`.slideDown()`和`.slideUp()`方法的用法与`.show()`和`.hide()`完全相同。选中某个元素，然后调用方法，一样有两个同样含义的可选参数（持续时间和回调函数）。不过这两个方法的名字常常会让jQuery新手犯糊涂。`.slideDown()`方法使元素由不可见变为可见，而`.slideUp()`则相反。

```
$('.message').slideDown();
```

5.4　使用`.slideToggle()`方法显示替代搜索项

搜索已成为网上不可或缺的一部分，很大程度上，谷歌正是依靠搜索成为今日世界知名的公

司。每个人进行搜索时，都期望能在你的站点上轻松找到他想要的东西。构建一个更好的用户界面，让用户能在弹指之间找到所有（他们想要的）东西，会是一个很重要的改进。Mozilla有一个庞大的开发者社区为Firefox浏览器编写扩展。图5-7是一个搜索栏，包括高级选项。当你单击高级选项，搜索栏的高级选择列表就滑下来，这是一个很好的体验，不用刷新页面就扩展了搜索选项。使用jQuery的`.slideToggle()`方法可以轻松实现这种效果。



图5-7　Firfox插件站点的高级搜索选项的滑动效果

`.slideToggle()`方法的用法与`.toggle()`非常相似，都用于显示和隐藏元素。假设目标元素现在可见，调用`.slideToggle()`则该元素就会滑上然后消失，否则该元素就滑下并显示出来。

```
$('.message').slideToggle();
```

下面这个解决方案中，我教你使用`.slideToggle()`创建一个支持滑动效果的高级搜索菜单，和图5-7中的例子类似。

(1) 准备搜索输入框及高级选项的HTML：

```
<style>
body {font-family:arial;}
.advanced {display:none;
   padding:3px;
   border:1px solid #ccc;
   width:300px;
}
```

```
</style>
<div id="search">
  <h1>Johnny's Superstore</h1>
  <input type="text" width="60" />
  <input type="submit" value="search"/><br>
  <a href="#" class="advanced-search">Advanced Search</a>
    <div class="advanced">
      <input type="radio" name="category"/> Clothing<br/>
      <input type="radio" name="category"/> Electronics<br/>
      <input type="checkbox" name="sale"/> Clearance Only<br/>
    </div>
  </div>
```

(2) 选中`.advanced-search`元素，并为它绑定`click`事件处理函数。在事件处理函数中，选中`.advanced`元素并调用`.slideToggle()`方法。

```
  $('.advanced-search').bind('click',function(){
    $('.advanced').slideToggle();
});
```


每单击一次Advance-Search（高级搜索）按钮，高级搜索选项就会根据页面加载后高级选项的状态滑上或者滑下。在本例中，我们使用CSS样式让高级选项元素在页面加载时隐藏。图5-8展示的是Firebug在Firefox浏览器中的输出。类似`.hide()`和`.show()`方法，jQuery通过给元素添加行内样式`display:block`让元素显示出来。

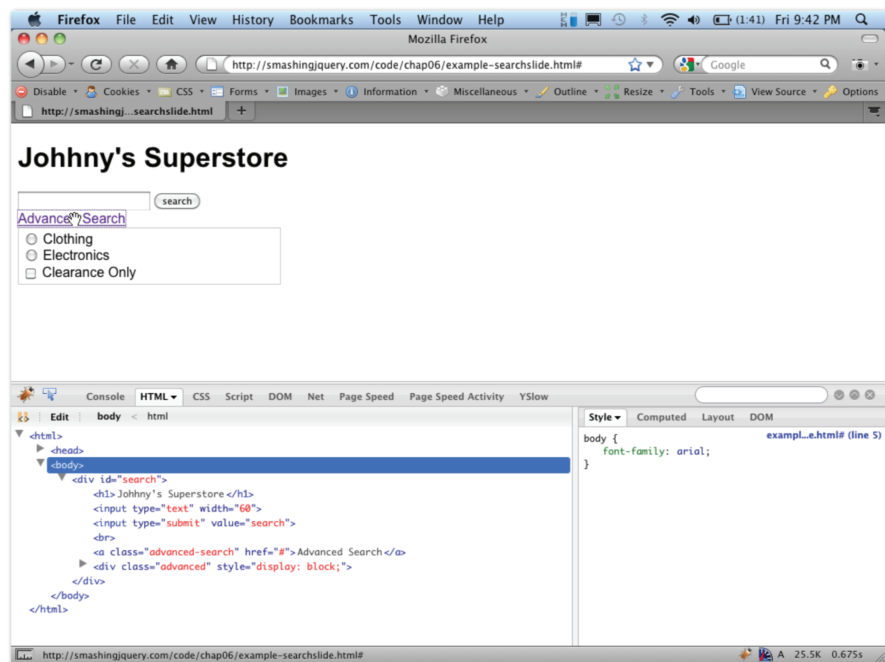


图5-8 Firebug在Firefox中的输出，类似`.show()`和`.hide()`，jQuery通过给元素添加行内样式`display:block`让元素显示出来

5.5　元素淡入淡出

淡入淡出效果使站点元素之间的交互又深入一层。淡入淡出这种过渡效果最常见于图库应用，或者图片相册。在这些应用中，一张图片淡出而另一张图片淡入。在几年前要想达到这种效果，如果不用Flash创建动画效果的图片相册，就只能使用高级JavaScript编写许多许多代码。

jQuery使得无需直接与原生JavaScript那难以理解的API打交道，就能用上JavaScript原生效果。多亏jQuery引进了这些效果，如今同样动画效果的图片相册才能够用JavaScript实现。使用JavaScript而不是Flash实现图片相册还有一个额外的好处，那就是你的应用对搜索引擎更加友好（SEO），因为不是所有的搜索引擎都能对Flash文件内的内容建立索引。

用于在网页上实现淡入淡出效果的一个关键CSS属性是`opacity`（见图5-9）。`opacity`属性的取值范围为0 ~ 100的整数值或者0.0 ~ 1.0的小数值，`.fadeIn()`和`.fadeOut()`方法[①]都使用了这个属性。



图5-9　图中图片正使用`opacity`属性淡出

`.fadeIn()`方法的用法与`.show()`方法一样。它也接受两个可选的参数——`duration`和`callback`。其中`duration`参数以毫秒（如600、200、700等）为单位，它决定动画速度（`fast/slow`）。`callback`参数是一个回调函数，该函数在特效结束后会立刻执行。

```
$(selector).fadeIn(duration, callback)
```

`.fadeOut()`方法与`.fadeIn()`方法几乎完全相同，不同的是它使选定元素淡出。

```
$(selector).fadeOut(duration, callback);
```

`.fadeTo()`方法允许我们指定目标元素淡入／淡出的目标透明度：

```
$(selector).fadeTo(duration, opacity, callback);
```

5.6　使用淡入／淡出效果建立一个简单的图库

为演示淡入淡出在站点上的应用，在本节我将从头构建一个简单的图库。图库中有5张图片，和一个可以单击以更换图片的数字列表。在更换图片时，当前图片淡出，选中的图片淡入。为便于讲解，我将脚本拆分成多个部分，这样你就能看清构建图库的每一个步骤，使脚本更灵活。

(1) 首先准备基础HTML。我使用jQuery添加图片相册必需的HTML，因此这个脚本非常容易安装部署：

① 还有`.fadeTo()`方法。

```
<div class="container">
  <h1>jQuery Images Galore.</h1>
</div>
```

(2) 接下来，准备图片库的样式表，确保图库在页面中的布局正确：

```
body {
  font-family:arial;
}

ul#nav {
  list-style-type:none;
  margin:10px 0 10px;
  padding:0;}

ul#nav li {
  float:left;
  width:30px;}

ul#nav li a {text-decoration:none;
   background:#05609A;
   color:#fff;
   padding:5px;}

ul#nav li a.active {
   background:#B4F114;
}

.slide-image {width:400px;
    height:300px;
    border:2px solid #05609A;
    overflow:hidden;
}

.slide-image img {
    display:none;
}
```

(3) 创建一个数组变量`slideArray`，存放所有图片的名字。我们会根据这个数组创建导航链接，有多少张图片，就创建多少个链接。我们随时可以根据需要调整这个数组元素的个数，脚本会自动地在页面上添加或减少图片。

```
var slideArray = [
  "ansel_adams1.jpg",
  "ansel_adams2.jpg",
  "ansel_adams3.jpg",
  "ansel_adams4.jpg",
  "ansel_adams5.jpg"
  ];
```

我们还需要创建一个变量`imgDir`[①]，保存图片相册中图片所在的目录，这个目录既可以是相对路径，也可以是绝对路径。在稍后准备好本解决方案中图片相册中的图片时，这个变量会被包

① 虽然这个变量名叫`imgDir`，作者也说它保存的是图片目录，但它存放的不仅仅是图片所在目录，还包括文件名的前缀。

5

含在（图片文件名）内。

```
var imgDir = 'images/ansel_adams';
```

(4) 在容器元素内添加图库元素。这个新添加的.slide-image元素用来存放稍后添加进来的全部图片。

```
$('.container').append('<div class="slide-image"></div>');
```

(5) 准备好.slide-image元素之后，我们需要在该元素中放一张图片，以便在页面加载完成之后显示它。

```
$('.slide-image').html('<img src="images/'+slideArray[0]+'"/>');
```

(6) 在.slide-image元素之后添加一个无序列表元素nav，用作导航。这个无序列表包含图片相册中所有图片的链接。

```
$('.slide-image').after('<ul id="nav"></ul>');
```

(7) 根据slideArray的length属性确定slideArray中一共有多少图片，并将它的值存放于变量slideLength当中。

```
var slideLength = slideArray.length;
```

(8) 构建一个for循环，迭代处理slideArray中的每一个项。该循环使用slideLength变量限制循环次数，在本例中循环体共运行5次。

```
for(i=0; i < slideLength; i++){
}
```

(9) 由于数组中共有5项，因此slideLength的值为5。我新建了一个变量slideText，并将它的值设置为 1 + i（循环变量）。这就确保了链接文本从1到5，而不是从0到4。

```
for(i=0; i <= slideLength; i++){
  var slideText = i + 1;
}
```

(10) 接下来我们利用第(5)步添加到DOM中的导航元素nav，为数组中的每一张图片生成一个li元素，并将它追加到#nav元素中。每个li中包含一个a标签，我们使用这个a标签的rel属性保存一份slideText，同时slideText也作为a元素的链接文本。我将利用rel属性来确定要插入哪张图片。图5-10显示的是在浏览器中执行循环结束后，使用Firebug看到的HTML结果。

```
for(i=0; i < slideLength; i++){
  var slideText = i + 1;
  $('#nav').append('<li><a href="#" rel="'+slideText+'">'+slideText+'</a></li>');
}
```

(11) 在创建导航列表之后，我们需要为每个链接元素绑定click事件。我们使用.bind()方法把事件处理函数绑定到导航链接元素的click事件：

```
$('#nav li a').bind('click', function(){
});
```

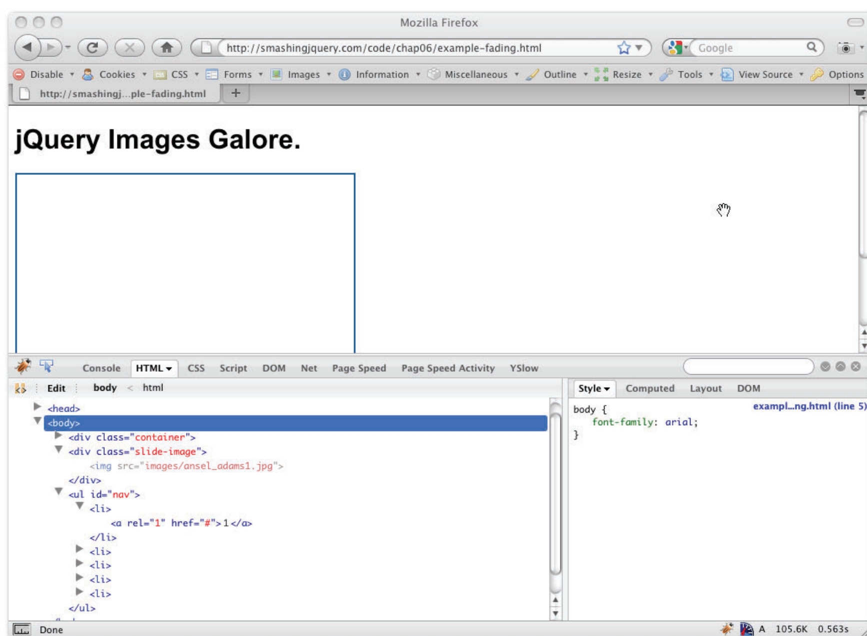


图5-10 循环结束后，使用Firebug看到浏览器中的HTML结果

(12) 新建一个变量numSlide，用它保存rel属性的值。这个变量当且仅当数字链接被单击时才会被赋值；换言之，赋值行为发生在事件发生时，而非绑定时。

```
$('#nav li a').bind('click', function(){
  var numSlide = $(this).attr('rel');
});
```

(13) 选中.slide-image元素，然后插入一个图片元素，并使用imgDir变量和numSlide变量拼出正确的图片src属性。这样，当click事件发生时，就能在.slide-image元素内添加上正确的图片。

```
$('#nav li a').bind('click', function(){
  var numSlide = $(this).attr('rel');
  $('.slide-image').html('<img src="'+imgDir + numSlide+'.jpg"/>');
});
```

(14) 当单击一个链接后，如果能告诉用户单击的是哪个链接，无疑会获得加分。为此我额外添加了两条语句。第一条移除导航菜单中所有链接的active类（上一次单击的链接），第二条语句为正被单击的链接添加active类。

```
$('#nav li a').bind('click', function(){
   var numSlide = $(this).attr('rel');
   $('.slide-image').html('<img src="'+imgDir + numSlide+'.jpg"/>');
   $('#nav li a').removeClass('active');
   $(this).addClass('active');
});
```

好了，现在你已经学会如何创建一个最简单的图库了。接下来我们添加淡入淡出效果，让这个图库像模像样。让图片淡入只需要一行代码。

(15) 选中.slide-image的图片子元素，然后调用.fadeIn()方法。

```
$('#nav li a').bind('click', function(){
  var numSlide = $(this).attr('rel');
  $('.slide-image').html('<img src="'+imgDir + numSlide+'.jpg"/>');
  $('.slide-image img').fadeIn();
  $('#nav li a').removeClass('active');
  $(this).addClass('active');
});
```

(16) 为确保页面加载完时显示第一张图片，选中导航的第一个链接，主动触发一次click事件：

```
$('#nav li a').eq(0).click();
```

5.7　使用延迟创建定时执行的动画

由于动画往往是在给定时间帧内发生的一系列事件，通过元素延迟创建定时执行的动画是一个很基础的需求。jQuery提供了一个.delay()方法，为延迟一段时间执行动画提供了一种解决方案。

只有1.4版本或更新的jQuery才有.delay()方法。它延迟指定的时间，然后再执行链接其后的方法。这个延迟方法仅适用于jQuery库的特殊效果。如果你需要一个更灵活的定时器函数，不妨试试JavaScript原生的setTimeout函数。

如果希望显示一条消息给用户，然后过上一段时间让消息自动消失，使用.delay()方法就非常合适。在下面这个例子里，我希望当用户将鼠标悬停在某个链接上时显示一条消息，当用户离开这个链接之后，等上10秒再让这条消息淡出。

(1) 准备（包着提示信息的）tooltip元素的HTML以及用于鼠标悬停的目标元素（鼠标悬停在该元素之上时显示提示信息）。

```
  <a href="#" class="show-tip">Learn more about Peaches</a>
      <div class="tool-tip">
      Although its botanical name Prunus persica suggests the peach is native to Persia,
peaches actually originated in China where they have been cultivated since the early
days of Chinese culture. Peaches were mentioned in Chinese writings as far back as the
10th century BC and were a favoured fruit of kings and emperors. Recently, the history
of cultivation of peaches in China has been extensively reviewed citing numerous
original manuscripts dating back to 1100 B.C.
      </div>
```

(2) 接下来，绑定hover事件处理函数。hover事件内部使用了mouseenter和mouseleave事件。第一个参数（回调函数）在mouseover事件发生时调用，选中.tool-tip元素，在900 ms的时间内采用淡入效果显示出来。

```
$('.show-tip').hover(
  function(){
  $('.tool-tip').fadeIn(900);
  },
  function() {
  });
```

(3) 第二个参数（回调函数）在mouseleave事件发生时调用。不过这次没有马上淡出，而是先延迟了10 000 ms（10 s）钟，然后用900 ms的时间淡出。

```
$('.show-tip').hover(
  function(){
    $('.tool-tip').fadeIn(900);
  },
  function() {
    $('.tool-tip').delay(10000).fadeOut(900);
  });
```

5.8 链式调用多个效果

现在你该很熟悉jQuery的链式调用了。链式调用允许在一条语句中顺序调用多个方法。这使得代码更少并且能提高脚本的性能。

在下面这个例子中,我使用链式调用语法演示在一个选择器语句中调用多个方法。.tool-tip元素先被隐藏，接着在900 ms的时间内淡入。经过1 s的延迟之后，它又花了900 ms淡出。

```
$('.tool-tip').hide().fadeIn(900).delay(1000).fadeOut(900);
```

如果不使用链式调用，就需要以下3条语句：

```
$('.tool-tip').fadeIn(900);
$('.tool-tip').delay(10000);
$('.tool-tip').fadeOut(900);
```

这3条语句的效果与前面链式调用的一条语句效果完全相同，但链式调用节省了代码，并且让代码更干净。设想这样一个场景：使用3个不同的元素ID选择3个li元素，并给它们设置不同的样式：

```
<ul id="news">
    <li id="politics">Politics</li>
    <li id="sports">Sports</li>
    <li id="finance">Finance</li>
    <li id="world">World</li>
    <li id="local">Local</li>
</ul>
```

有两种方式完成这件事。第一种，如下所示，写3条语句，每次选中一个元素并设定样式：

```
$('#politics').css('border','1px solid red');
$('#finance').css('display','none');
$('#local').css('border','1px solid green');
```

第二种则利用.end()方法实现所有元素和方法的链式调用。.end()方法能得到包含上一步操作结果集的jQuery对象，从而让其后链接的方法不受.end()函数之前选择器的影响。图5-11展示的是用链式调用[①]重写上面代码后的运行结果[②]。

```
$('#news').find('#politics').css('border','1px solid red').end().find
 ('#finance').hide().end().find('#local').css('border','1px solid blue');
```

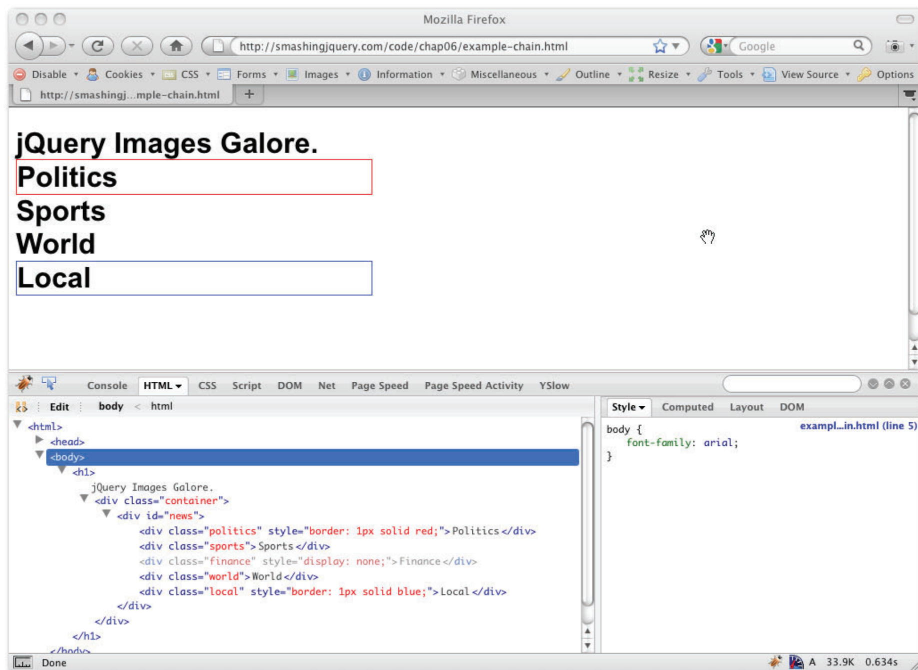


图5-11　用链式调用方法重写上面的代码示例后的最终结果

5.9　使用多种特效创建一个新闻阅读器

随着通过RSS投递新闻内容日渐普及，新闻阅读器这样的小工具越来越常见。新闻阅读器既可以是一个可在页面重新加载时随时刷新显示10篇文章的方框，也可以是相当高级的类似谷歌实时搜索引擎（图5-12）那样支持实时更新的新闻阅读器。

① 下面这个例子中的链式调用有点刻意而为的意思。其实就这个例子来说，分开写代码更清楚，代码量也更少。译者认为链式调用是jQuery提供的福利，如果链式调用写出来更清楚，我们就用链式调用，否则，我们就分开写。

② 虽然默认看不到，但jQuery会在jQuery对象中缓存上一步的结果，以备不时之需。

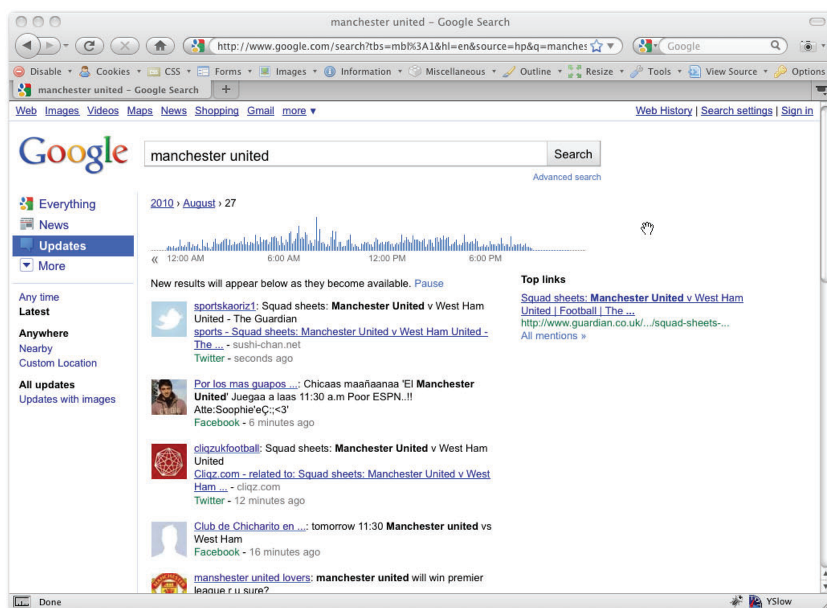


图5-12　谷歌实时搜索引擎（来自Twitter的新内容随时滑动显示出来）

下面我来做一个抓取静态内容的简单新闻阅读器，这个阅读器使用了淡入和滑动特效。我们可以扩展这个新闻阅读器以便支持实时RSS feed或者做一些优化，不过现在这个简单版本用来教学已经足够了。

(1) 第一步，生成基本的HTML结构。我们的目标是让新闻阅读器尽量灵活，这样它就能够放到任意一个页面而无需修改页面。在本例中，我先创建了一个带h1元素的页面，稍后我会使用选择器把新闻阅读器的代码扔到这个页面中去。

```
<body>
  <h1>jQuery Latest News</h1>
</body>
```

(2) 我在数组变量newsArray中存放了10个文章标题。新闻阅读器会遍历这些静态内容。

```
var newsArray = [
"Delhomme, Wallace sharp early for Browns",
"Bucs expect to have injured QB Freeman for opener",
"Report: Haynesworth likely has rhabdomyolysis",
"QB Orton effectively leading Broncos in preseason",
"Vernon Gholston not offended by set-up fight",
"Cubs' Piniella to retire after Sunday",
"Bradley interested in Aston Villa job",
"Federer beats Fish for Cincinnati title",
"Garcia 3-hits Giants, Cardinals roll 9-0",
"Cano, CC power Yankees over M's 10-0"
];
```

(3) 创建两个变量：newsLength保存文章数量，也就是newsArray的长度；newsInterval变量存放一个以毫秒为单位的数值，它定义过多长时间抓取新的文章标题并放到新闻阅读器中。

```
var newsLength = newsArray.length;
var newsInterval = 2000;
```

(4) 选中h1元素，并在它后面添加一个无序列表news-feed。

```
$('h1').after('<ul id="news-feed"></ul>');
```

(5) 生成一个for循环，遍历newsArray中的标题。针对每一个标题，添加一个li元素来包裹这个标题。

```
for(i=0; i < newsLength; i++){
$('#news-feed').append('<li>'+newsArray[i]+'</li>');
}
```

(6) 接下来，编写一个slideHeadline()函数，它包含所有的特效，让新闻阅读器可用。这个函数体的第一条语句选中无序列表news-feed中的最后一个li元素，克隆它，再使用.prepend()方法把克隆得到的li元素添加到新闻列表的最前头（并让它隐藏）。

```
function slideHeadline() {
  $('#news-feed li:last').clone().prependTo('#news-feed').hide();
}
```

(7) 函数体的第二条语句选中上一步克隆的元素，然后调用.slideDown()方法让它以滑动方式显示出来。

```
function slideHeadline() {
  $('#news-feed li:last').clone().prependTo('#news-feed').hide();
  $('#news-feed li:first').slideDown();
}
```

(8) 我希望第一个条目用500 ms的时间滑下，还希望它用1000 ms的时间以淡入方式显示出来。我在包含slideDown方法调用的同一条语句中利用链式调用添加.fadeIn()。

```
function slideHeadline() {
  $('#news-feed li:last').clone().prependTo('#news-feed').css('display','none');
  $('#news-feed li:first').fadeIn(1000).slideDown(500);
}
```

(9) slideHeadline函数的最后一条语句用于删除列表中的最后一项。这3条语句一条接一条的执行，如图5-13所示，极好地模拟了滑动和淡入效果。

```
function slideHeadline() {
  $('#news-feed li:last').clone().prependTo('#news-feed').css('display','none');
  $('#news-feed li:first').fadeIn(1000).slideDown(500);
  $('#news-feed li:last').remove();
}
```

(10) 最后一句JavaScript代码至关重要。我需要利用原生JavaScript函数setInterval以固定的时间间隔（2000 ms）不断执行slideHeadline函数。没有这个函数，新闻阅读器无法工作。

```
setInterval(slideHeadline, newsInterval);
```

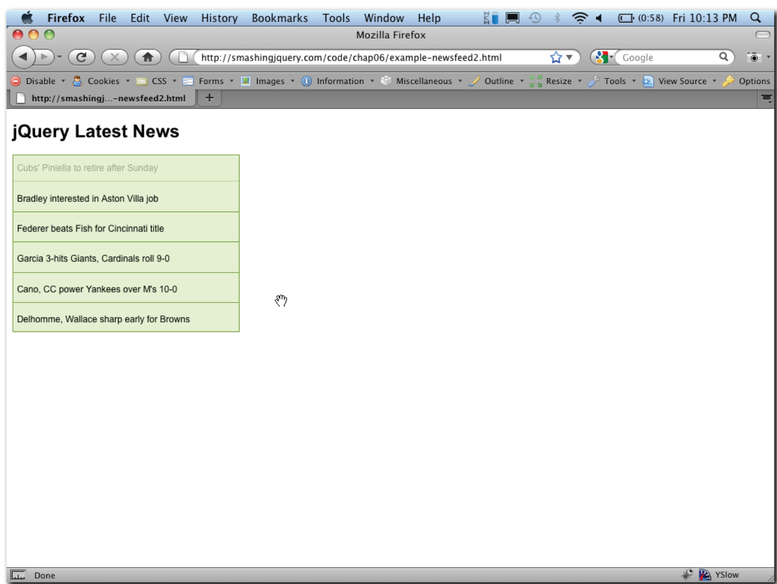


图5-13　顶部不断有新元素淡入，同时底部元素不断被删除的效果图

`setInterval`函数是一个原生JavaScript定时器函数。它允许我们指定一个函数和一个时间间隔，时间一到，函数立即执行。这一点类似于另一个JavaScript定时器函数`setTimeout`，二者最大的区别在于`setInterval`在你通知它停止前会不间断地（隔上指定的时间）执行指定的函数，而`setTimeout`只执行一次。`clearTimeout()`函数用来取消`setTimeout()`产生的定时器，`clearInterval()`则用来取消`setInterval()`产生的定时器。jQuery库正是使用`setTimeout`和`setInterval`函数才实现了其异常强大的特效API。

5.10　创建高级动画

jQuery有一个`.animate()`方法，我们可以利用它生成自定义动画。与链式调用`fade`系列、`slide`系列和`show`等作用范围有限的方法不同，`.animate()`方法允许我们使用任意的可通过数值动态控制的CSS属性。表5-2列出了`.animate()`方法可用的CSS属性。

表5-2　可用于**`.animate()`**方法的常见CSS属性

CSS属性	示　例　值
`opacity`	`0.5`
`top`	`10 px`
`height`	`100 px`
`width`	`200 px`
`margin`	`10 px`
`padding`	`15 px`

5.10.1　使用高级动画创建一个带文本说明的图库

我以前一直使用Flash和ActionScript，大约两年前决定改变方向，开始更多地使用JavaScript和jQuery。之所以会做这样的决定，有两个原因，一个是JavaScript对DOM的控制能力越来越强，另一个是一些移动平台如iPhone不支持Flash和ActionScript。Flash非常擅长处理高级动画，不过基于JavaScript图库的应用已经取得长足进展。在学习ActionScript的过程中，我最大的收获是发现这两种语言（JavaScript和ActionScript）在处理事件和特效上相当一致。[①]

理解编程语言背后的基本原理对学习一门新语言有极大的帮助，因为绝大多数概念都是一样的；有变化的仅仅是语言的语法。ActionScript 3和JavaScript都基于ECMAScript，因此它们有许多相似之处。由于JavaScript的语法和习惯与ActionScript如此接近，在使用JavaScript的过程中我不止一次有“熟识”之感。

2006年，我使用Flash 8编写了自己的第一个由XML驱动的动画效果的图片相册。那是一个相当灵活的小应用。你提供一个格式化好的XML给Flash文件，它就迭代处理这个XML，添加图片，然后以动画方式展示它们。每一张图片都支持单击，能把你带往一个新页面（参见图5-14）。这是我非常得意的一个小应用。有多得意？我把它发布到了个人博客onerutter.com，它一共被人们下载了13 000多次。这个Flash文件大小约5 KB，一共使用了136行代码。那个时候这已经非常短小精悍了，足以令人印象深刻。直到开始使用jQuery，我才明白那些代码还是太多了。



图5-14　我2006年的作品——一个用Flash 8写成的基于XML的图库应用

① 这毫不奇怪，JavaScript和ActionScript都遵守ECMAScript规范，可以视为同一家族的两个兄弟。

下面这个图片相册案例教你编写一个图片相册，它与我2006年用Flash编写的应用非常相似，不过不是使用XML驱动，且只有大约85行代码。如果你需要，可以扩展这个程序，添加XML数据源支持（我把这个任务留给你）。图5-15展示的是这个图片相册的工作方式。

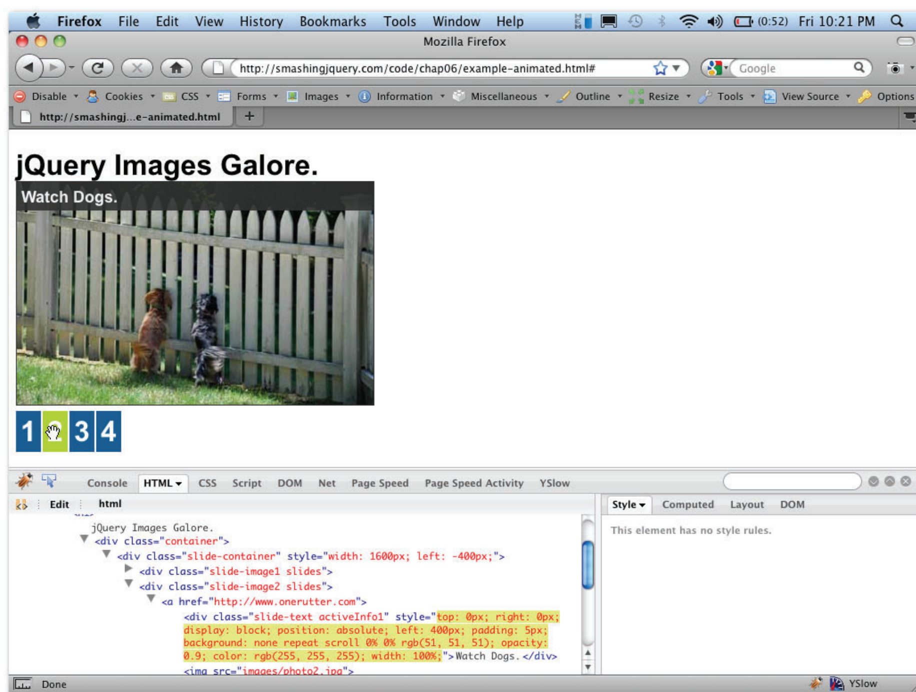


图5-15　图片相册的工作方式

(1) 页面上需要添加的唯一一个HTML元素是`div.container`，所有其他HTML都由脚本生成，填充到这个容器中。

```
<div class="container"></div>
```

(2) 接下来准备图片相册的样式表。如果你使用的图片和我使用的图片大小不一样，就修改`.container`的属性以适应你的尺寸。当图片切换时，`.text-strip`类用来设置图片上方的文本样式。

```
body {font-family:arial;}

ul#nav {
  list-style-type:none;
  margin:10px 0 10px;
  padding:0;}

ul#nav li {
  float:left;
  width:30px;}
```

```
ul#nav li a {text-decoration:none;
  background:#05609A;
  color:#fff;
  padding:5px;}

ul#nav li a.active {
  background:#B4F114;}

.container {position:relative;
  height:250px;
  width:400px;
  border:1px solid #333;
  overflow:hidden;}

.slide-container {
  position: absolute;
  top: 0;
  left: 0;}

.slides {
  float:right;}

  .slide-text {
  display:none;
  font-size:18px;}

img {border:0;}

.textStrip {top:0px;
  display:block;
  position:absolute;
  left:-400px;
  padding:5px;
  background:#333333;
  opacity:.9;
  color:#ffffff;
  width:100%;
}
```

(3) 首先我们要创建3个数组。第一个数组`slideArray`，用来存放图片文件的名字，这些图片将作为图片相册中的一张张幻灯片显示在图片相册中。第二个数组`textArray`，用来保存每张图片的说明。第三个数组`urlArray`，用来保存每张图片对应的链接（URL）。你需要在图片相册中显示几张图片，就在这些数组中准备同样多的元素。

```
var slideArray = ["photo1.jpg","photo2.jpg","photo3.jpg","photo4.jpg"];
var textArray = ["Rusty Cable.","Watch Dogs.","Plant Sink.","Urban Cowboy"];
var urlArray = ["http://www.google.com", "http://www.onerutter.com", "http://
  www.flickr.com", "http://www.facebook.com"];
```

(4) 将`div.slide-container`追加到页面上的`div.container`元素中。顾名思义，它用来存放所有的图片。

```
$('.container').append('<div class="slide-container" />');
```

(5) 在slide-container元素之后，插入一个无序列表ul#nav。这个无序列表用来保存导航链接，控制显示图片相册中的哪一张图片。它还有一个clearfix属性，负责设定导航列表之后的CSS clear属性（清除浮动）。

```
$('.slide-container').after('<ul id="nav" class="clearfix"></ul>');
```

(6) 编写一个for循环，循环次数为slideArray数组的长度。slideArray有4个元素，因此最大的索引值是3。在循环体内新建一个变量slideNum，并将它的值设置为 i + 1。之所以如此设置是因为i的第一个值是0，而数组中第一张图的文件名中的编号是1，如果不如此设置，slideNum与文件名就无法匹配。

```
for(i=0; i < slideArray.length; i++){
var slideNum = i + 1;
}
```

(7) 循环体的第二条语句用来往ul#nav元素中追加li元素及相应的值。

```
for(i=0; i < slideArray.length; i++){
    var slideNum= i + 1;
    $('#nav').append('<li><a href="#" rel="'+slideNum+'">'+slideNum+'</a></li>');
}
```

(8) 接下来，添加一个变量slideInfo，用它来存放显示图片的HTML。HTML代码由好多行组成，如果每一行之后有额外的空白，JavaScript脚本就不能工作。为避免此问题，我们将HTML分成多个子字符串并用+=运算符连接起来。这样代码既干净又易读。

第一条语句创建一个类名前缀为slide-image的元素，在它的名字中使用了slideNum变量，因此每张图片的类名都不相同。第二条语句添加一个类名为slide-text的div元素，它的内容为与图片对应的说明。第三条语句添加循环变量i对应图片的标签，它的src值来自slideArray。

```
for(i=0; i < slideArray.length; i++){
    var slideNum = i + 1;
    $('#nav').append('<li><a href="#" rel="'+slideNum+'">'+slideNum+'</a></li>');
    var slideInfo = '<div class="slide-image'+slideNum+' slides">';
    slideInfo += '<div class="slide-text">'+textArray[i]+'</div>';
    slideInfo += '<img src="images/'+slideArray[i]+'"/></div>';
}
```

(9) HTML拼接完成之后，把它追加到.slide-container元素中。

```
for(i=0; i < slideArray.length; i++){
    var slideText = i + 1;
    $('#nav').append('<li><a href="#" rel="'+slideText+'">'+slideText+'</a></li>');
    var slideInfo = '<div class="slide-image'+slideText+' slides">';
    slideInfo += '<div class="slide-text activeInfo'+[i]+'">'+textArray[i]+'</div>';
    slideInfo += '<img src="images/photo'+slideText+'.jpg"/></div>';
    $('.slide-container').append(slideInfo);
}
```

(10) 接下来再添加3个变量。slideTotal保存图片的总数，也就是slideArray的长度4。slideWidth是每张图片的宽度，而slideContainer则是宽度乘以slideTotal。在本例是中slideContainer的值是1600。

5

```
var slideTotal = slideArray.length;
var slideWidth = 400;
var slideContainer = slideWidth * slideTotal;
```

(11) 利用CSS方法，使用上一步中创建的变量slideContainer的值重新设定.slide-container元素的宽度。

```
$(".slide-container").css({'width' : slideContainer});
```

(12) 为导航#nav无序列表的子元素li的后代元素a绑定click事件处理函数。

```
$('#nav li a').bind('click', function(){
});
```

(13) 当链接被单击时高亮显示相关项，这样就能明确标识用户刚刚单击的是哪一张图片。要允许当前项高亮显示，我们需要增加两条语句。第一条删除所有元素的active类。第二条语句使用this关键字为刚刚单击的元素添加active类。

```
$('#nav li a').bind('click', function(){
$('#nav li a').removeClass('active');
  $(this).addClass('active');
});
```

(14) 如果此时.slide-text动画尚未结束，click事件处理函数中接下来的这条语句会使用.css()方法重设.slide-text元素的位置。

```
$('#nav li a').bind('click', function(){
$('#nav li a').removeClass('active');
$(this).addClass('active');
$(".slide-text").css({
  'top':'-100px',
  'right':'0px'
});
```

(15) 上一条语句重新设定.slide-text元素的位置。接下来的两条语句使用.stop()方法和.clearQueue()方法停止动画并清理动画序列。如果不清理动画序列，.stop()方法虽然停止了动画，但动画未完成的部分会进入动画序列，造成不想要的结果。

```
$('#nav li a').bind('click', function(){
$('#nav li a').removeClass('active');
$(this).addClass('active');
$(".slide-text").css({
  'top':'-100px',
  'right':'0px'
  });
$(".slide-text").stop();
$(".slide-text").clearQueue();
});
```

(16) 创建更多的变量。第一个是active，它的值来自导航当前活跃元素的rel属性值减1。第二个变量slideNum，保存导航当前活跃元素的rel属性的值（不减1）。第三个变量slidePos，其值等于active乘以slideWidth。

```
$('#nav li a').bind('click', function(){
  $('#nav li a').removeClass('active');
  $(this).addClass('active');
  $(".slide-text").css({
    'top':'-100px',
    'right':'0px'
  });
  $(".slide-text").clearQueue();
  $(".slide-text").stop();

  var active = $('#nav li a.active').attr("rel") - 1;
  var slidePos = active * slideWidth;
  var slideNum = $('#nav li a.active').attr("rel");
});
```

(17) 在上一步中新建的变量中，如果active的值等于2，而slideWidth的值等于400，则slidePos的值等于800。在这个例子里，对于选中.slide-container元素并应用.animate()方法让图片向左移这件事情，slidePos变量至关重要。我们在调用.animate()方法时还提供参数duration（值1000）以及一个回调函数。


```
$('#nav li a').bind('click', function(){
$('#nav li a').removeClass('active');
$(this).addClass('active');
$(".slide-text").css({
  'top':'-100px',
  'right':'0px'
});
$(".slide-text").clearQueue();
$(".slide-text").stop();

var active = $('#nav li a.active').attr("rel") - 1;
var slidePos = active * slideWidth;
var slideNum = $('#nav li a.active').attr("rel");

$(".slide-container").animate({
  left: -slidePos,
  },1000, function(){
});
});
```

(18) 我们在第一条animate语句的回调callback函数中设置.slide-text的动画效果。使用唯一的.slide-text类选中.slide-text元素，然后使用.css()方法设定必要的样式，以便.slide-text可以显示在当前图片之上。

```
$('#nav li a').bind('click', function(){
    $('#nav li a').removeClass('active');
    $(this).addClass('active');
    $(".slide-text").css({
        'top':'-100px',
        'right':'0px'
    });
    $(".slide-text").stop();
```

```
        $(".slide-text").clearQueue();
        var active = $('#nav li a.active').attr("rel") - 1;
        var slidePos = active * slideWidth;
        var slideNum = $('#nav li a.active').attr("rel");
        $(".slides-container").animate({
            left: -slidePos
        },1000, function(){
            $('.slide-image'+ slide Num+' .slide-text').addClass('textStrip');
        });
    });
```

(19) 在最后的回调函数中，再添加一个`.animate()`方法调用，在一秒钟内将`.slide-text`移动到当前图片之上，然后过一会儿再让`.slite-text`从图片上滑动消失。图5-16显示了该脚本在浏览器中的最终效果。

```
    $('#nav li a').bind('click', function(){
      $('#nav li a').removeClass('active');
      $(this).addClass('active');

      $(".slide-text").css({
      'top':'-100px',
      'right':'0px'
      });

      $(".slide-text").stop();
      $(".slide-text").clearQueue();

      var active = $('#nav li a.active').attr("rel") - 1;
      var slidePos = active * slideWidth;
      var slideNum = $('#nav li a.active').attr("rel");

      $(".slide-container").animate({
      left: -slidePos
      },1000, function(){
        $('.slide-image'+slideNum+' .slide-text').addClass('textStrip').animate({
            top:0,
            left:slidePos,
            right:0
        }, 1000, function(){
          $('.slide-text').delay(5000).animate({
          top:-100
          }, 1000);
           });
       });
    });
```

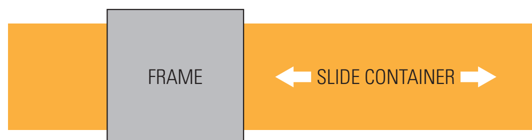


图5-16　该脚本在浏览器中的最终效果

5.10.2 使用jQuery Easing插件添加缓动效果

由GSGD（http://gsgd.co.uk/sandbox/jquery/easing/）提供的jQuery Easing插件为jQuery动画增添了30种缓动特效。.animate()方法内建支持两种缓动特效——swing和liner，只是这两种效果非常有限，要生成类似颤动或弹跳等更逼真的动画效果，最好还是使用插件。在动画过程中通过加速／减速控制动画的缓动效果，尤其常见于一个动画戛然而止时。缓动特效使动画效果更逼真。

```
.animate(duration, easing, callback);
```

jQuery可通过插件扩展功能（我们已经知道，可以编写自己的jQuery函数），如果对于某个具体任务希望能够重用其实现代码或者打算把它发布到jQuery开源社区供别人使用，我们就可以把它改编成一个jQuery插件。在第11章我会详细介绍jQuery插件。

下面是Easing插件支持的一些缓动效果：

- easeOutBounce
- easeInBounce
- easeInElastic
- easeInCubic

easeOutBounce和easeInBounce特效是一对颤动特效组合，用来模拟一个物体在屏幕上弹跳。easeInElastic特效让一个元素像松紧带那样啪的一声弹回就位。easeInCubic类似传统的渐快特效，不过要慢很多。Easing插件的站点上有所有30种缓动特效的演示效果。

下载这个缓动插件，然后在页面中把它包含在jQuery库后，就可以使用了。出于演示目的，我要为上面的案例添加一个缓动效果。

从上面的例子中抽出以下代码。如果我们把easing参数设置为easeOutBounce，动画就会使用自定义缓动插件提供的easeOutBounce特效，就这么简单！

```
$(".slide-container").animate({
    left: -slidePos
},'easeOutBounce',1000, function(){
    $('.slide-image'+slideNum+' .slide-text').css({
        'display':'block',
        'position':'absolute',
        'top':'0px',
        'left':'-400px',
        'padding':'5px',
        'background':'#333333',
        'opacity':'.9',
        'color':'#ffffff',
        'width':'100%'
    }).animate({
        top:0,
        left:slidePos,
        right:0
```

5

```
    },'easeOutBounce',1000, function(){
        $('.slide-text').delay(5000).animate({
            top:-100
        }, 1000);
    });
});
```

Part 3

第三部分

jQuery 应用

本部分内容

- 第 6 章　改进导航：菜单、标签及折叠选项
- 第 7 章　生成可交互的生动表格
- 第 8 章　使用 jQuery 制作高级表单

第 6 章

改进导航：菜单、标签及折叠选项

页面导航帮助你穿梭于各个页面之间，寻找产品，阅读每日新闻或者登录网上银行查看自己的收支情况。在网上使用导航已经成为许多用户的第二本能。利用前几章学到的jQuery事件和特效，能有效地增强用户体验。

改进导航和交互方式可提高客户满意度，从而提高网站访问量。我将在本章介绍一些难用导航的真实案例，并教你使用jQuery、HTML和CSS改进它们。

6.1　让页面上所有的链接都在新窗口打开

能一次选取多个元素并修改这些元素是jQuery的一个诱人功能。有时你希望页面上所有的链接都在新窗口打开，但没有修改页面源代码的权限。页面通常由某个地方的PHP、JSP（Java Server Pages）或ASP（Active Server Pages）文件生成，而你无权访问它们。

不过，我们有jQuery。无需打开一大堆文件逐个修改比对，只需要一丁点儿jQuery代码就可解决这个问题，这既可靠又节省时间。使用jQuery的好处在于，当我们不再需要这个功能时，只需注释掉这部分代码或者干脆删除它。

现在我来教你使用jQuery设置页面上所有的链接属性，让它们改在新窗口中打开。除此之外，我们还可以对页面上的这些链接做些别的事情，比如：

- 给所有的链接添加一个新的类；
- 为所有链接添加内容与链接文本一致的`title`属性；
- 为所有的链接分别添加一个`rel`属性；
- 删除所有链接的`href`属性，从而禁用这些链接。

在下面这个解决方案中，我会写一段代码，示范如何遍历给定列表中的所有链接，为它们添加`title`属性，并让它们改在新窗口中打开（见图6-1）。

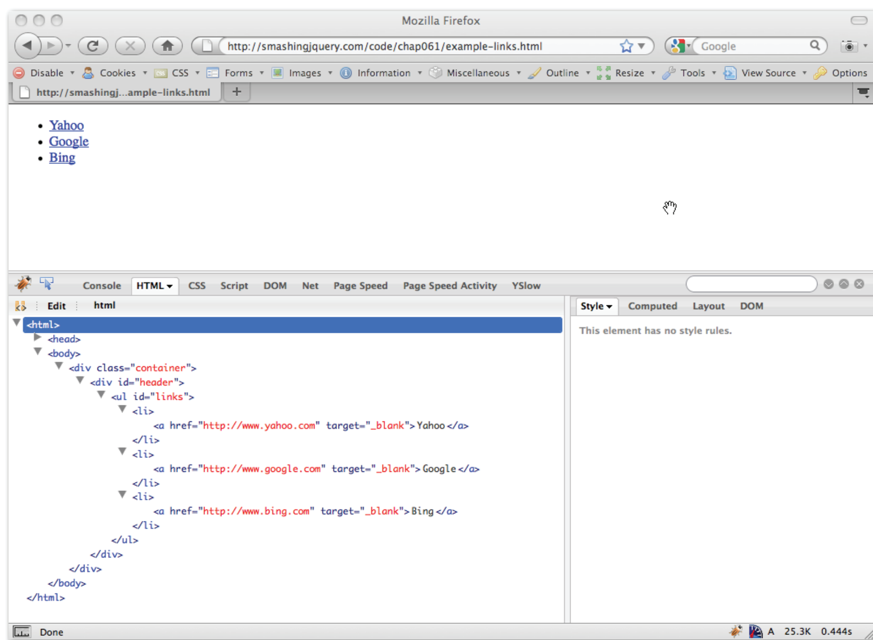


图6-1　Firefox中打开的Firebug窗口，清晰地显示链接的target属性已变为_blank

(1) 本例中的HTML是一个无序列表ul#links。我们将利用它的ID选取列表中的链接。

```
<ul id="links">
  <li><a href="http://www.yahoo.com">Yahoo</a></li>
  <li><a href="http://www.google.com">Google</a></li>
  <li><a href="http://www.bing.com">Bing</a></li>
</ul>
```

(2) 选取#links li元素的后代元素a，然后使用.attr()方法把这些链接元素的target属性设置为_blank。

> .attr()方法可以用来设置或者获取属性的值。

```
$('#links li a').attr('target', '_blank');
```

6.2　突出显示导航中的当前选中项

任何时候用户都希望能在导航菜单中明确地看到自己“身处何处”。然而这种基本的用户界面需要却常常被忽视。图6-2展示的就是这样一个例子：你已经导航到了一个特定的小节，但主导航上的菜单项却仍然（傻乎乎地）保持高亮。就这个问题来说，我见过一些jQuery和JavaScript解决方案，不过更常见的解决办法是使用后端语言，如PHP、JSP或ASP。

下面的解决方案带你使用jQuery设置导航菜单，让当前选中的菜单项保持高亮。jQuery脚本对菜单项URL与页面实际URL进行比对，让正确的菜单项高亮。

菜单项仍然（错误地）保持高亮



由Mashable.com许可复制

图6-2　即使已导航到一个特定的小节，主菜单上的菜单项仍保持高亮

(1) 创建一个变量path，并为它赋值为location.pathname，pathname是JavaScript原生对象location的一个属性，它返回URL中域名之后的部分。

```
var path = location.pathname;
```

假设我们正在访问页面

www.smashingjquery.com/mycode/myexample.html。那么location.pathname就等于/mycode/myexample.html。

(2) 再创建一个变量pathArray，然后使用原生的Java Script字符串方法.split()把pathname从/处拆分，得到一个数组。

```
var pathArray = path.split('/');
```

本例中pathArray的值是['mycode', 'myexample.html']。也就是/mycode/myexample.html的拆分结果。

(3) 创建最后一个变量pArrLength，把它的值设置为pathArray的长度。

```
var pArrLength = pathArray.length;
```

(4) 创建一个for循环，迭代处理数组pathArray的每一个值。然后使用属性选择器选中包含其中某个值的所有链接，为其加上selected类。这里的属性选择器使用了通配符（*），它会匹配href的任意部分。[①]

① 这里的匹配很粗放，生产环境下建议严格匹配整个URL。

```
for(i=0; i < pArrLength; i++) {
  $("a[href*='"+pathArray[i]+"']").addClass("selected");
}
```

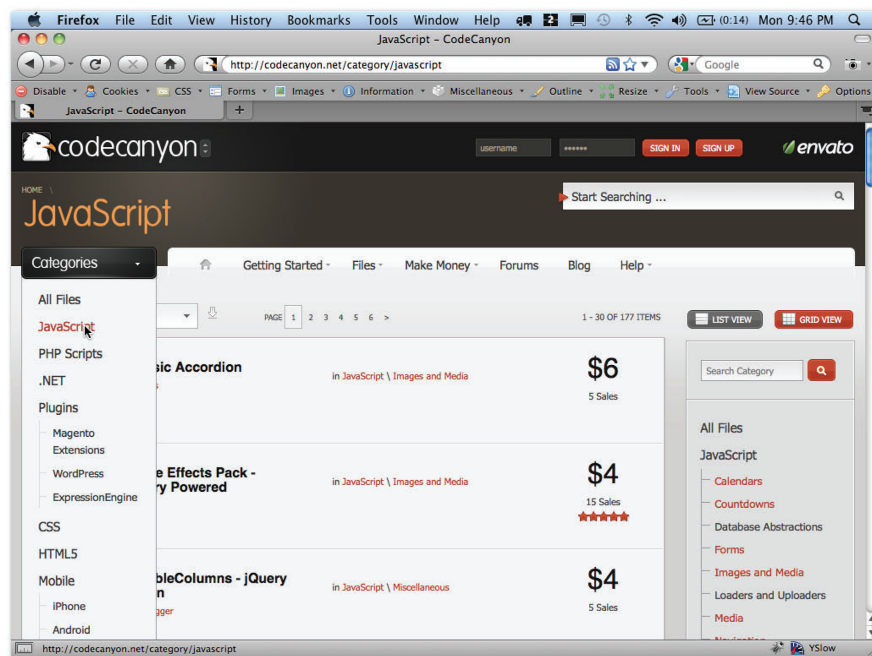
这个高亮菜单项脚本具有一些局限性，它只适用于没有下拉菜单的导航。如果有下拉菜单，我们不仅要高亮当前菜单项，还要高亮它的父菜单项，因此需要修改这个脚本，以适应下拉菜单。举例来说，如果顶级菜单项是“关于我们”，它的下拉项子菜单有“我们的故事”、“工作机会”和“联系我们”，我们必须构建可能的URL集合并使用`.parent()`和`.children()`等选择器匹配这个集合。

6.3　创建简单的下拉菜单

下拉菜单是页面导航最流行的方案之一。许多站点都把功能分成几组，每组设置一些页面。要建立一个极其详细的、一次展示网站所有页面的导航系统，不但在技术上是个挑战，甚至有可能把用户吓跑。

下拉菜单的好处在于我们能够在菜单中列出所有页面，但默认只显示最顶级的菜单。然后由用户自己来决定（把鼠标悬停在顶级菜单上）是否需要查看更多的项。

下拉菜单可以水平摆放，也可以垂直摆放在主菜单之下（参见图6-3）。



由envato.com许可复制

图6-3　下拉菜单示例

图6-4所示是一个简单的下拉菜单。我要教你编写这个菜单，完成之后再给它加上一点装饰（参见图6-5）。

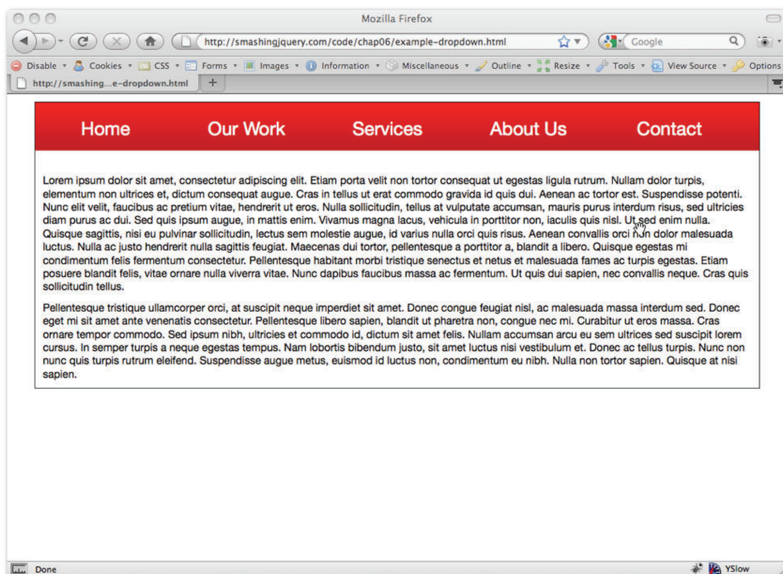


图6-4　设计在浏览器中的效果

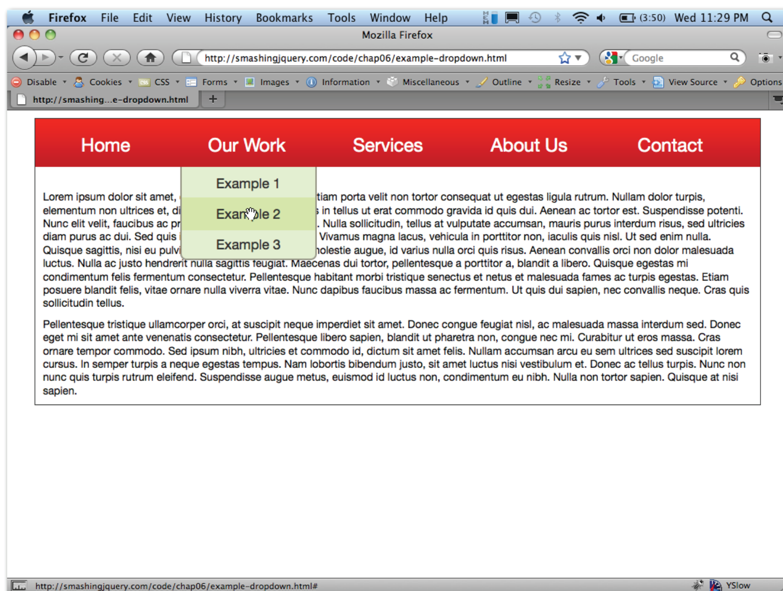


图6-5　演示mouseenter事件怎样“反冒泡”到子元素（另见彩插图6-5）

(1) 下拉菜单由嵌套的无序列表构成，这样的设计非常灵活，可以随意添加菜单层次和菜单项，而无需担心会破坏设计。下面的代码中，我创建了5个顶级菜单链接和2个下拉菜单（分别位于Our Work和Services之下）。你自己的菜单项可多可少，视具体需要而定。在需要菜单的地方加上这些菜单项，小心别忘记关闭列表的标签。

```
<div id="header">
  <ul id="navigation">
    <li><a href="#">Home</a></li>
    <li><a href="#">Our Work</a>
      <ul class="subnav">
        <li><a href="#">Example 1</a></li>
        <li><a href="#">Example 2</a></li>
        <li><a href="#">Example 3</a></li>
      </ul>
    </li>
    <li><a href="#">Services</a>
      <ul class="subnav">
        <li><a href="#">Service 1</a></li>
        <li><a href="#">Service 2</a></li>
        <li><a href="#">Service 3</a></li>
      </ul>
    </li>
    <li><a href="#">About Us</a></li>
    <li><a href="#">Contact</a></li>
  </ul>
</div>
```

(2) 接下来准备样式表，这样才能保证菜单在页面中显示正常。我总是在样式表的开头放上清理样式（CSS reset），且最近一直在用YUI（Yahoo! User Interface）清理样式，它真是好用极了。

CSS清理样式表负责把有关样式的值重置为0，既让所有的浏览器在渲染页面时公平竞争，也让页面更容易兼容多个浏览器。在清理样式之后，你必须为这些样式设置新的值（以0为基准）。由于本例使用了清理样式，有关浏览器默认设置已经重置，无需为兼容各个浏览器增加许多额外的CSS样式，从而节省了大量的时间。

我把清理样式也放到了这里，如果你需要，尽管拿去用。

```
body, div, dl, dt, dd, ul, ol, li, h1, h2, h3, h4, h5, h6, pre, form, fieldset, input,
  textarea, p, blockquote, th, td{margin:0; padding:0}
h1, h2, h3, h4, h5, h6{font-size:100%; font-weight:normal}
```

(3) 剩下的CSS控制页面上主菜单和下拉菜单的布局。在开始jQuery编程等基础工作之前，我们先完成所有的CSS，把下拉菜单的表现层处理好。

```
.container{
    width:950px;
    margin:10px auto;
    font:14px "Helvetica Neue",Arial,Helvetica,Geneva,sans-serif;
    border:1px solid #333;
}
p{margin:10px;}
```

6

```
ul#navigation{
   list-style-type:none;
   background:#CE0100;
   height:63px;
   font-size:24px;
}

ul#navigation li{
   float:left;
   width:175px;
   text-align:center;
   position:relative;
   height:63px;
   padding:20px 5px 10px 5px;
}

ul#navigation li a{
   color:#fff;
   text-decoration:none;
   display:block;
}

ul#navigation li a.active{
   border:1px solid blue;
}

ul#navigation li ul.subnav{
   background:#E7F1D2;
   width:175px;
   clear:both;
   display:none;
   position:absolute;
   top:63px;
   -moz-border-radius-bottomleft:8px;
   -moz-border-radius-bottomright:8px;
   -webkit-border-bottom-left-radius:8px;
   -webkit-border-bottom-right-radius:8px;
   border-radius: 8px;
   border-left:2px solid #998;
   border-right:2px solid #998;
   border-bottom:2px solid #998;
}

ul#navigation li ul.subnav li{
   clear:both;
   height:40px;
   padding:0;
   text-align:center;
   margin:0px;
}
```

```
ul#navigation li ul.subnav li a{
   background:none;
   font-size:18px;
   color:#333;
   text-decoration:none;
   padding:10px 0;
   border:none;
}

ul#navigation li ul.subnav li a:hover{
   background:#DBF1AD;
   font-size:18px;
   color:#333;
   border:none;
}
```

(4) 接下来，我们使用选择器#navigation li选中菜单项，设置它的hover事件。hover事件是mouseenter和mouseleave事件的组合。第一个事件处理函数处理mouseenter事件，我们需要用.find()找出类名为subnav的嵌套列表，然后调用.slideDown()方法把它显示出来。第二个事件处理函数处理mouseleave事件，它做的事情和mouseenter事件处理函数相反，即用.find()找出相应的子菜单，然后调用.slideUp()方法把它收起来。对那些没有子菜单（.subnav）的元素来说，当鼠标悬停在它们之上时，由于找不到.subnav子元素，什么都不会发生。

6

.find()方法帮助我们从结果集中过滤出符合选择器的后代元素，得到并返回匹配结果集。

```
$(document).ready(function(){
  $('#navigation li').hover(function() {
    $(this).find('.subnav').slideDown('slow');
  }, function() {
    $(this).find('.subnav').slideUp('fast');
  });
});
```

你或许会想，为什么不像下面这样，直接找出.subnav元素并让它显示出来？

```
$('.subnav').slideDown('slow');
```

上面这条语句没有限定条件（this），它会匹配所有的.subnav元素并将它们显示出来，这显然是错误的。先选中#navigation li元素，然后绑定hover事件，我们就可以用this关键字引用被hover的元素，从而确保.slideDown()方法只应用到被悬停列表项的子菜单上。

(5) 最后一步，当鼠标位于主菜单或相应的子菜单上时，主菜单和相应的子菜单应该保持高亮。图6-5展示的是mouseenter事件从父元素“反冒泡”到子元素的一个例子。使用.find()方法找到对应的a元素，如果是mouseenter事件就添加active类，否则就删除active类。

```
$(document).ready(function(){
  $('ul#navigation li').hover(function() {
    $(this).find('.subnav').slideDown('slow');
    $(this).find('a').addClass('active');
```

```
    }, function() {
      $(this).find('.subnav').slideUp('fast');
      $(this).find('a').removeClass("active");
    });
  });
```

一旦我们有了一个能正常工作的下拉菜单，用jQuery给菜单添加交互就非常简单了。你可以花大量的时间自己做或者和设计师一起做美化菜单和设计的工作。图6-6展示的是Firefox浏览器输出，其中Firebug插件已开启，可观察到DOM中下拉菜单代码的变化。

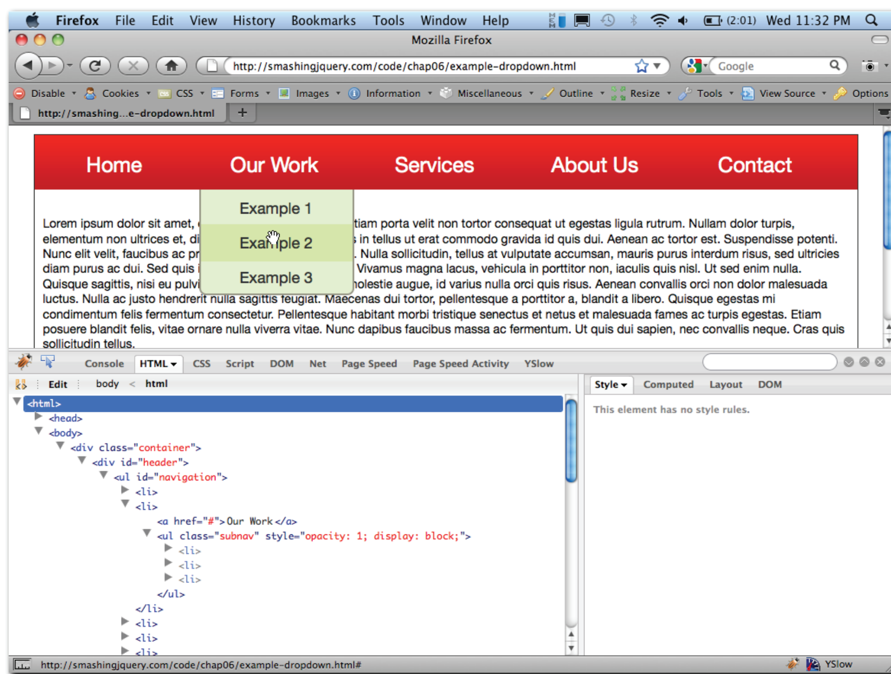


图6-6　Firefox浏览器中Firebug插件已开启，可观察到DOM中下拉菜单代码的变化（另见彩插图6-6）

使用.animate()方法为下拉菜单增加特效

在本节中，我将教你在菜单下拉时添加透明度变化。我们将使用.animate()方法为上一节中创建的下拉菜单添加高级动画效果。我们延用上一个例子的全部代码，只需要修改两行jQuery代码。.animate()方法让我们能够控制菜单下拉过程中的更多细节。我们需要给它4个参数：CSS属性、动画持续时间、缓动方式及回调函数。在这个具体例子中，我希望切换height属性并改变透明度，把动画持续时间设置为500 ms。你也可以将fast（相当于200 ms）或者slow参数（相当于600 ms）用作参数。结果如图6-7所示。

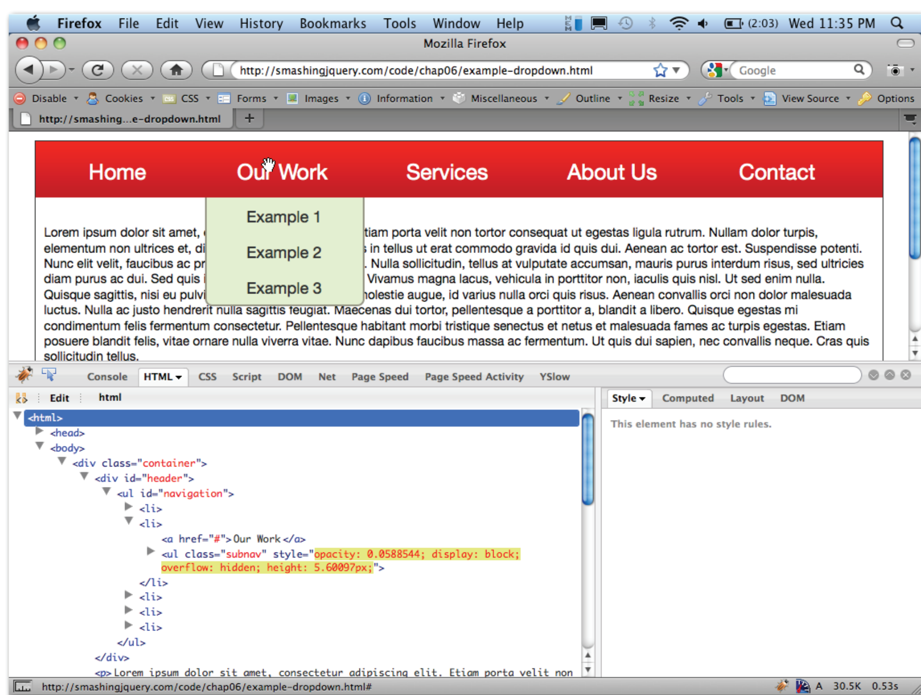


图6-7　Firefox中看到的应用于下拉菜单的高级特效

我们在两条语句中使用.animate()方法切换滑动（下拉/卷起）特效，在hover事件发生时把opacity和height这两个CSS属性参数传递给.animate()方法。

```
$(document).ready(function(){
  $('ul#navigation li').hover(function() {
    $(this).find('.subnav').animate({opacity: 1.0,height: 'toggle'}, 500);
    $(this).closest('a').addClass('active');
}, function() {
    $(this).find('.subnav').animate({opacity: 0,height: 'toggle'}, 500);
    $(this).find('a').removeClass("active");
                });
  });
```

6.4　创建折叠菜单

折叠菜单是很流行的设计。当需要在一小块地方显示一大堆内容时，Web设计师和UI设计师常常会选用折叠菜单。折叠菜单只显示一节的内容，同时隐藏其他内容。当鼠标悬停在某项上或者单击一个具体的项时，相关联的内容就展开或缩起来。折叠菜单可用于导航，不过最近它们广泛用于在一小块地方显示大量的内容等与显示内容相关的场合。Magento（开源商务平台）就实现了一个折叠选项风格的结账程序，如图6-8所示。

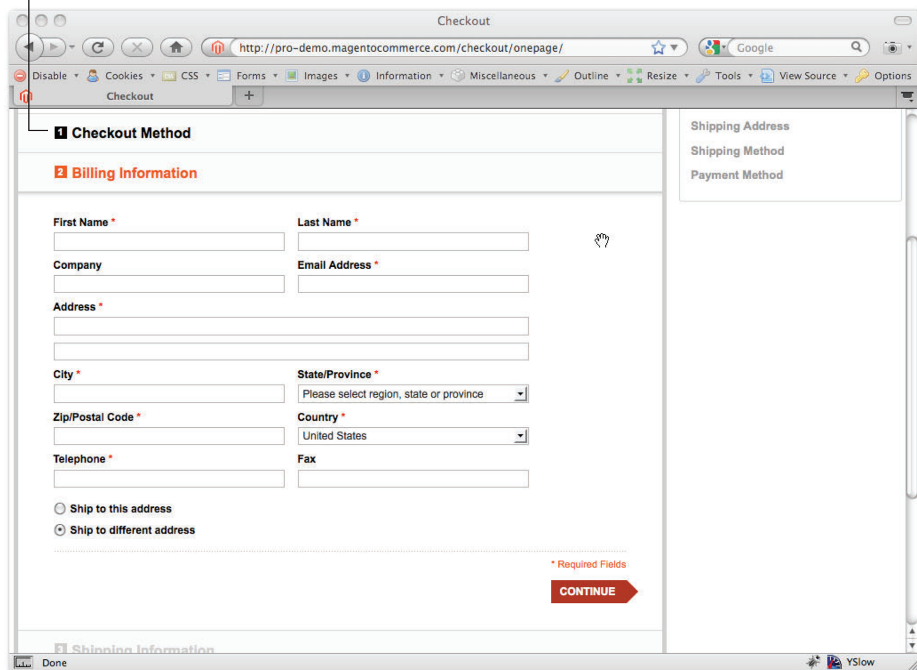


图6-8 Magento站点使用了折叠选项风格的结账程序

在这一节的案例中，我们一起实现一个用于网站侧栏导航的垂直折叠菜单。折叠菜单（同一时刻）只有一个小节是展开状态，其余各节都呈关闭状态。标题元素会显示一个图标，指示当前展开的是哪个小节。

(1) 首先定义折叠菜单的HTML结构。我们的目标是创建一个便于jQuery操作和维护的干净利落的结构。HTML结构需要两个元素，一个是折叠菜单标题元素，一个是折叠菜单内容元素。其中标题元素有一个accordion-header类，它包裹着一个用来显示标题文本的h3标签。单击标题元素可显示或隐藏其下方的内容。内容元素是一个无序列表，具有accordion-content类。单击标题时，内容元素包裹着的内容就会视情况隐藏或显示。

```
<div id="accordion">
  <div class="accordion-header">
    <h3>Books</h3>
    <span></span>
  </div>

  <ul class="accordion-content">
    <li><a href='#'>Business</a></li>
    <li><a href='#'>Education</a></li>
    <li><a href='#'>Tech</a></li>
```

```
    <li><a href='#'>Romance</a></li>
  </ul>

  <div class="accordion-header">
    <h3>Electronics</h3>
    <span></span>
  </div>

  <ul class="accordion-content">
    <li><a href='#'>Audio</a></li>
    <li><a href='#'>Video</a></li>
    <li><a href='#'>Automobile</a></li>
    <li><a href='#'>Appliances</a></li>
  </ul>

  <div class="accordion-header">
    <h3>Sporting Goods</h3>
    <span></span>
  </div>

  <ul class="accordion-content">
    <li><a href='#'>Baseball</a></li>
    <li><a href='#'>Basketball</a></li>
    <li><a href='#'>Football</a></li>
    <li><a href='#'>Tennis</a></li>
  </ul>
</div>
```


(2) 接下来，我们需要设置样式让折叠菜单在页面中正常显示。和前面的解决方案类似，我们先加入CSS清理样式：

```
body, div, dl, dt, dd, ul, ol, li, h1, h2, h3, h4, h5, h6, pre, form, fieldset, input,
  textarea, p, blockquote, th, td{margin:0; padding:0}
h1, h2, h3, h4, h5, h6{font-size:100%; font-weight:normal}
```

(3) 余下的CSS控制折叠菜单的外观。我们给菜单和内容元素以不同的样式，菜单项包含一个图标，用以指示当前哪个菜单项被选中。

菜单图标使用了CSS“精灵”（sprite）。“精灵”是指创建一张大图，然后通过背景定位技术用这一张图为Web站点上的多个元素提供背景。使用“精灵”最大的好处是可大大减少站点向Web服务器请求图片的次数，从而降低了载入时间，提高了站点性能。如果你的站点中使用了20多张图片，就能明显体会到使用“精灵”时的性能改善。此外，“精灵”还有助于防止图片载入过程中的闪烁（比如使用两张独立的图片制作翻转效果，并且被悬停的图片尚未加载完时）。

```
#accordion{
   width:225px;
   margin:10px 0 10px 10px;
}

#accordion .accordion-header{
   background:#3971AC;
   color:#fff;
```

```
    border-bottom:1px solid #fff;
    position:relative;
}

#accordion .accordion-header h3{
    margin:0;
    cursor:pointer;
    text-indent:10px;
    padding:5px 0;
}

#accordion .header-active{
    background:#48ABC3;
}

#accordion .accordion-header span{
    background:url(../images/accordion_sprite.gif) no-repeat;
    display:block;
    position:absolute;
    width:11px;
    height:12px;
    top:5px;
    left:200px;
}

#accordion .accordion-header span.icon-active{
    background:url(../images/accordion_sprite.gif) no-repeat;
    background-position:0 -12px;
    display:block;
    position:absolute;
    width:11px;
    height:12px;
    top:5px;
    left:200px;
}

#accordion ul.accordion-content{
    margin:0px 0 0px 0;
    padding:5px 5px 10px 5px;
    list-style-type:none;
    background:#A8D7E2;
}

#accordion ul.accordion-content li{
    padding:1px 0px;
    display:block;
    margin:0;
    padding:2px 5px;
}

#accordion ul.accordion-content li a{
    color:#D16C3A;
}
```

(4) 在页面加载完成之后，确保只有第一个菜单元素及其内容元素可见。以下jQuery代码结合使用.not()方法和 :first 过滤器，使第一个元素之外的其他折叠内容（内容元素）隐藏。

```
$('.accordion-content').not(':first').hide();
```

.not()方法过滤掉参数选择器匹配的任意元素或者选择器。参数选择器可以是某种过滤器（如本例所示），也可以是一个具体的元素（在下面这一步我们就会看到这种例子）。

(5) 隐藏完除第一个外的所有.accordion-content元素之后，我们用另一条语句找出第一个.accordion-content元素，并调用.show()方法（确保它可见）：

```
$('.accordion-content:first').show();
```

(6) 为了让当前菜单项高亮，我在CSS文件中定义了一个header_active类。使用下面的代码把这个类应用到第一个.accordion-header元素：

```
$('.accordion-header:first').addClass('header-active');
```

我还使用空的span标签为折叠菜单的标题加了图标。我定义了一个icon-active类，它会显示成一个向下的箭头，提示用户这个内容区是展开状态。如果标题没有这个类， span标签就会显示成一个向右的箭头。使用:first过滤器选中第一个.accordion-header元素，然后链式调用.find()方法找出它的span后代元素，为它加上.icon-active类。

```
$('.accordion-header:first').find('span').addClass('icon-active');
```

6

(7) 现在进入有趣的部分——让折叠菜单响应用户输入，也就是为.accordion-header标题元素绑定click事件处理函数。

```
$('.accordion-header').click(function () {
});
```

(8) click事件处理函数中的第一条语句负责让当前展开的元素缩回去，并移除使它们高亮的类。为此，只需选中可见的.accordion-content元素，并调用.slideUp()。之后再调用.prev()得到DOM树的上一个元素，也就是对应的.accordion-header元素，删掉该元素的header-active类。

```
$('.accordion-header').click(function () {
  $('.accordion-content:visible').slideUp('slow').prev()
    .removeClass('header-active');
});
```

.prev()方法用于查找DOM树中上一个兄弟元素。.prev()方法还支持选择器参数，比如你可以将.active作为参数传递给它。

(9) 增加一条语句，确保把所有可见的.icon-active元素显示成向右的箭头，方法是移除icon-active类。

```
$('.accordion-header').click(function () {
  $('.accordion-content:visible').slideUp('slow').prev()
    .removeClass('header-active');
  $('.icon-active:visible').removeClass('icon-active');
});
```

请务必记住，.addClass()、removeClass()和hasClass()方法的参数，类的名字前不需要句点。这是一个常被忽视的问题，会导致jQuery脚本错误。另外，.is()、.filter()方法和.not()方法在参数为类名时，一定要加上句点。

(10) 增加一条语句，使用this关键字选中被单击的菜单标题，给它加上header-active类。然后用.next()方法选中它的下一个元素，即内容元素，调用.slideDown()让其慢慢滑下。

```
$('.accordion-header').click(function () {
  $('.accordion-content:visible').slideUp('slow').prev()
    .removeClass('header-active');
  $('.icon-active:visible').removeClass('icon-active');
  $(this).addClass('header-active').next().slideDown('slow');
});
```

.next()方法与.prev()方法正相反，它不返回上一个兄弟元素，而是返回下一个。

(11) 最后一条语句使用this关键字，结合.find()方法找到.accordion-header元素中的span标签，给它加上icon-active类。

```
$('.accordion-header').click(function () {
  $('.accordion-content:visible').slideUp('slow').prev()
    .removeClass('header-active');
  $('.icon-active:visible').removeClass('icon-active');
  $(this).addClass('header-active').next().slideDown('slow');
  $(this).find('span').addClass('icon-active');
});
```

如果在站点或应用中，折叠菜单项不是页面固有的，而是被动态添加到DOM中的，你一定要改用.live()方法绑定click事件。否则那些在DOM加载完之后才添加进来的元素不会响应click事件。

将所有的HTML、CSS和jQuery代码组合到一起，并在浏览器中载入这个页面，你就会看到一个生龙活虎的折叠菜单（参见图6-9）。

折叠菜单脚本能帮助我们在一小块地方显示大量的内容，不过它也有很多局限性，如下。

- 随着不断向折叠菜单中添加菜单项，我们开始需要滚动页面才能选择或查看其中嵌入的内容。如果是在一个真实项目中使用折叠菜单，这对用户界面来说是相当大的缺陷，它会降低界面的可用性，如果是一个电子商务站点的话，甚至会降低销售额和转化率。
- 刷新页面时，折叠菜单并不会记住上一次展开的是哪个菜单项，不过我们可以通过jQuery和cookie实现这类功能。解决这个问题还有一项更好的、不使用cookie的技术，那就是在URL的查询字符串中使用锚，详见Rebecca Murphey的一篇博文（http://blog.rebeccamurphey.com/2007/12/04/anchor-based-url-navigation-with-jquery/）。
- 折叠菜单的当前位置无法保存到书签中。如果你通过添加书签收藏一个页面，期望下次打开它时能打开保存时看到的小节，你的期望会落空。这个问题可以通过上一段中提到的锚技术解决。
- 如果你的站点受众缺乏因特网经验，他们有可能不熟悉折叠菜单，因此很可能根本就不会用。

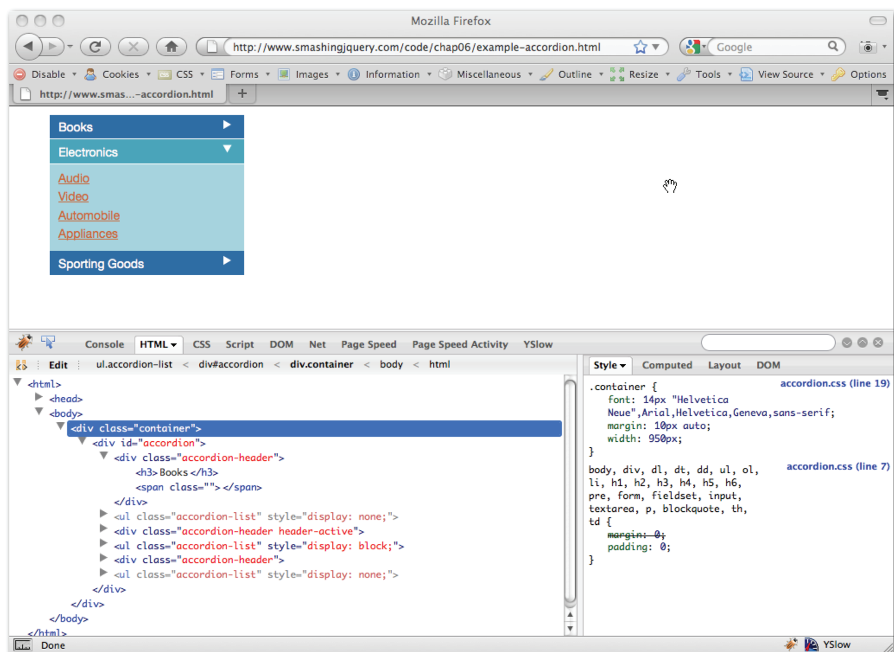


图6-9　Firefox中的折叠菜单脚本，在Firebug面板上一切都清清楚楚

6.5　创建标签式内容

标签导航不但容易实现，而且非常直观，深受Web设计师和开发者的喜爱，常被用于Web站点或应用内的导航。标签结构不禁使人回想起在文件柜中使用悬挂标签文件夹区分文件内容的年代。如今标签结构已经被成功移植到Web上，成为一种把数据组织到逻辑“文件柜”或网页的极佳方式。图6-10展示的是Basecamp项目管理工具，它的工具箱（tools）部分采用了标签来组织和显示内容。

标签式导航不仅用于网页，也常见于计算机桌面。一些浏览器，比如Mozilla Firefox和Google Chrome，使用标签在一个窗口中显示多个Web页，最近IE7也采用了这种用户界面。苹果机上的聊天程序Adium和PC上的Trillian也采用了标签式导航，用于在一个窗口中组织多个聊天会话。

一些站点结合动态内容切换技术把标签式导航推向了另一个高度。这时标签并非用于站点中不同页面之间的导航，而是用于在同一页面动态切换显示内容，这与折叠菜单在一个较小的区域显示较多的内容有点儿类似，却更容易理解和使用。一些站点更进一步，当单击标签时才使用Ajax技术载入要显示的内容，显示的内容实际上来自服务器而非页面，而且能够做到每一次都显示不同的内容，这才是真正的动态内容切换。图6-11展示的是Coda站点（www.panic.com/coda），它不但在页面中使用标签展示动态内容，而且在呈现内容时使用了动画技术，当用户单击标签时，内容不是从左边滑入，就是从右边滑入，相当漂亮。

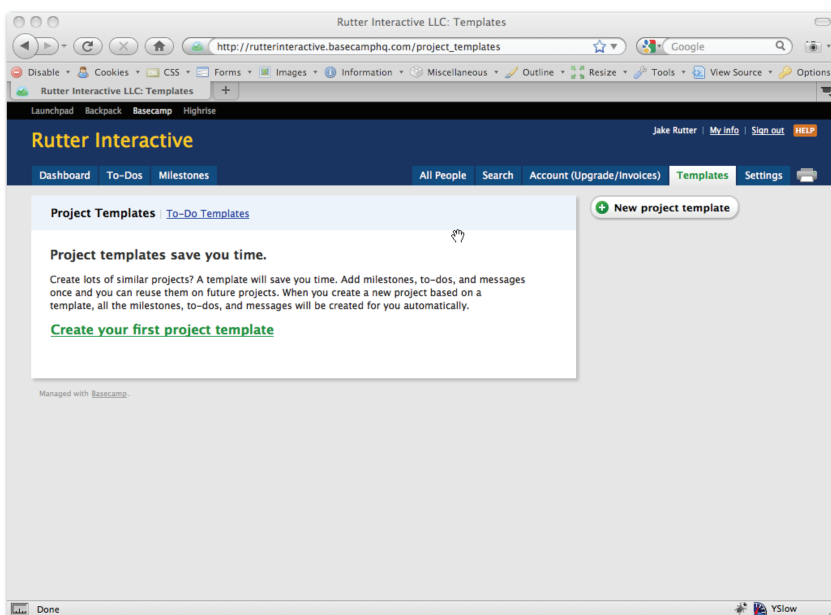


图6-10　Basecamp项目管理工具，它的工具箱部分采用了标签来组织和显示内容



© 2007，PANIC公司（Panic、Panic Logo和Transmit为Panic公司注册商标；www.panic.com）

图6-11　Coda站点在页面中使用标签显示动态内容

在下面这个教程中，我带领大家使用标签实现一个动态内容切换页面。这个教程中的jQuery脚本非常健壮，支持随时添删内容和标签。

(1) 定义标签的HTML结构。标签需要两个元素——导航元素和内容元素。`#tabs`无序列表用作标签，其中的每个`li`元素内有一个链接，链接文本即标签的显示内容。在本教程中，我使用了以下5个导航标签：

```
<ul id="tabs" class="clearfix">
  <li><a href="">Home</a></li>
  <li><a href="">Shop</a></li>
  <li><a href="">Community</a></li>
  <li><a href="">Customer Service</a></li>
  <li><a href="">About</a></li>
</ul>
```

(2) 添加标签内容。每一个标签对应的内容包装在一个名为`.content`的元素中。你可以添加更多或更少的内容标签，只要数量和标签的个数一致就行。`.content-container`元素包裹着所有的标签。在本例中，`.content-container`包含以下5部分内容：

```
<div id="content-container">
<div class="content">
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer euismod nunc id
mauris placerat iaculis. Integer viverra velit eros, sed semper ante. Ut at turpis in
tellus tincidunt dignissim non vitae felis. Donec nec sem ut est tincidunt ullamcorper.
</div>

<div class="content">
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer euismod nunc id
mauris placerat iaculis. Integer viverra velit eros, sed semper ante. Ut at turpis in
tellus tincidunt dignissim non vitae felis. Donec nec sem ut est tincidunt ullamcorper.
</div>

<div class="content">
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer euismod nunc id
mauris placerat iaculis. Integer viverra velit eros, sed semper ante. Ut at turpis in
tellus tincidunt dignissim non vitae felis. Donec nec sem ut est tincidunt ullamcorper.
</div>

<div class="content">
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer euismod nunc id
mauris placerat iaculis. Integer viverra velit eros, sed semper ante. Ut at turpis in
tellus tincidunt dignissim non vitae felis. Donec nec sem ut est tincidunt ullamcorper.
</div>

<div class="content">
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer euismod nunc id
mauris placerat iaculis. Integer viverra velit eros, sed semper ante. Ut at turpis in
tellus tincidunt dignissim non vitae felis. Donec nec sem ut est tincidunt ullamcorper.
</div>

<div class="content">
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer euismod nunc id
```

```
mauris placerat iaculis. Integer viverra velit eros, sed semper ante. Ut at turpis in
tellus tincidunt dignissim non vitae felis. Donec nec sem ut est tincidunt ullamcorper.
</div>
</div>
```

(3) 接下来，我们添加重置样式以确保标签在各浏览器中显示一致：

```
body, div, dl, dt, dd, ul, ol, li, h1, h2, h3, h4, h5, h6, pre, form, fieldset, input,
  textarea, p, blockquote, th, td{margin:0; padding:0}
h1, h2, h3, h4, h5, h6{font-size:100%; font-weight:normal}
.clearfix:after{content:".";display:block;height:0; clear:both; visibility:hidden}
```

(4) 剩下的CSS负责让标签和内容各就各位。记住一个关键点，所有的`.content`元素默认都是隐藏的。

```
body{background:#8CCAD9}

ul#tabs{
   list-style-type:none;
   position:relative;
}

ul#tabs li{
   float:left;
   width:155px;
   text-align:center;
   position:relative;
   margin:0 3px;
   position:relative;
}

ul#tabs li a.tab-active{
   color:green;
   background:#fff;
   position:relative;
   top:1px;
}

ul#tabs li a{
   border-top:1px solid #9B4C24;
   border-left:1px solid #9B4C24;
   border-right:1px solid #9B4C24;
   background:#2E7D91; ;
   padding:10px 5px 10px 0px;
   display:block;
   text-decoration:none;
   font:bold 14px "Helvetica Neue",Arial,Helvetica,Geneva,sans-serif;
   color:#fff;
   position:relative;
}

#content-container{
   border:1px solid #333;
   background:#fff;
}

.content{display:none;}
```

以下所有jQuery代码都要包在document.ready()事件处理函数内，确保在DOM准备好之后，才执行我们的jQuery代码。

(5) 页面加载完成之后，我们需要让第一个标签的内容显示出来。下面这条语句使用:first过滤器选中.content元素，然后调用.show()方法将它显示出来。

```
$('.content:first').show()
```

(6) 在第一个内容元素显示出来之后，我们还要选中第一个导航标签（#tabs li a:first），为它加上tab-active类，从而保证页面加载完成之后，第一个标签呈选中状态。

```
$('#tabs li a:first').addClass('tab-active');
```

(7)为导航元素（#tabs li a）绑定hover事件。hover事件综合了mouseenter和mouseleave事件，这个hover事件完全是可选的，添加它纯粹是为了加特效。

```
$("#tabs li a").hover(
  function () {
  // mouseenter 事件
  },
  function () {
  // mouseleave 事件
  }
);
```

6

(8) 由于动画效果当且仅当鼠标悬停在标签之上时才显示出来，所以我们应该在mouseenter事件处理函数中添加一条调用.animate()方法的语句。当用户把鼠标放到标签上300 ms之后，标签就在50 ms（持续时间）之内向左移动20 px，然后弹回原处。如果你更喜欢其他效果，可以添加其他CSS属性来调整.animate()。

.animate()方法只支持属性值为数值的CSS属性，如margin、padding、left、top和opacity。

```
$("#tabs li a").hover(
  function () {
  // mouseenter 事件
    $(this).animate({left:20}, 300, function (){
      $(this).animate({left:0}, 50);
    });
  },
  function () {
  // mouseleave 事件
  }
);
```

(9) 为导航元素（#tabs li a）绑定click事件处理函数。这个click事件处理函数控制页面中内容元素的显示和隐藏。我们先添加一个return false语句阻止单击事件的默认行为（重定向到被单击的链接#）。

```
$('ul#tabs li a').bind('click',function () {
  return false;
});
```

(10) 然后编写click事件处理函数真正的第一条语句，获取被单击标签的索引号（第几个），并把它保存到变量linkIndex：先选中所有的导航标签（#tabs li a），然后调用.index()方法，并以this关键字作为.index()的参数，将结果赋值给变量linkIndex。

```
$('ul#tabs li a').bind('click',function () {
  var linkIndex = $('#tabs li a').index(this);
  return false;
});
```

使用索引号而非一个静态的类或者ID名，让我们的脚本支持动态添加或删除标签等HTML更改操作，从而避免像那些硬编码的脚本一样，一旦遇到标签数量变化，就要修改脚本的情况。如果这个脚本支持任意数量的标签，它会具有更大的价值，因为它更容易扩展。

当为.index()方法传入一个参数（选择器或元素），它计算并返回该对象的位置号。

(11) 再往click事件处理函数中加一条语句，选中所有导航标签（#tabs li a），移除tab-active类。这条语句保证了同一时刻只有一个标签高亮。

```
$('ul#tabs li a').bind('click',function () {
  var linkIndex = $('ul#tabs li a').index(this);
  $('#tabs li a').removeClass('tab-active');
  return false;
});
```

(12) 再加一条语句到click事件处理函数，选中当前可见的内容元素，调用.hide()方法将其隐藏。

```
$('ul#tabs li a').bind('click',function () {
    var linkIndex = $('ul#tabs li a').index(this);
    $('ul#tabs li a').removeClass('tab-active');
    $(".content:visible").hide();
    return false;
});
```

(13) 再加一条语句到click事件处理函数，根据被单击标签的索引号找到相应的内容元素，将其显示出来。该语句依据元素的索引号控制内容元素的显示和隐藏。尽管DOM中标签元素（li）和内容元素之间并无关系，当单击标签时，却可以通过标签的索引号找出它对应的内容元素。这是一种极好的保持代码良好移植性的方法。

```
$('ul#tabs li a').bind('click',function () {
  var linkIndex = $('ul#tabs li a').index(this);
  $('ul#tabs li a').removeClass('tab-active');
  $(".content:visible").hide();
  $(".content:eq("+linkIndex+")").show();
  return false;
});
```

:eq(index)过滤器返回结果集中索引号index对应的元素。

(14) 在click事件处理函数中再加最后一条语句，为被单击的标签加上tab-active类，让用户知道当前显示的是哪个标签。

```
$('ul#tabs li a').bind('click',function () {
  var linkIndex = $('ul#tabs li a').index(this);
  $('ul#tabs li a').removeClass('tab-active');
  $(".tab:visible").hide();
  $(".tab:eq("+linkIndex+")").show();
  $(this).addClass('tab-active');
  return false;
});
```

如果你的站点或应用程序使用脚本动态载入标签及其对应的内容，就要改用live方法绑定click事件。否则，那些在页面加载完成之后才添加到页面的新元素，不会绑定click事件。

将所有的HTML、CSS和jQuery代码组合到一起，并在浏览器中载入这个页面，你就会看到一个生动的标签导航（参见图6-12）。

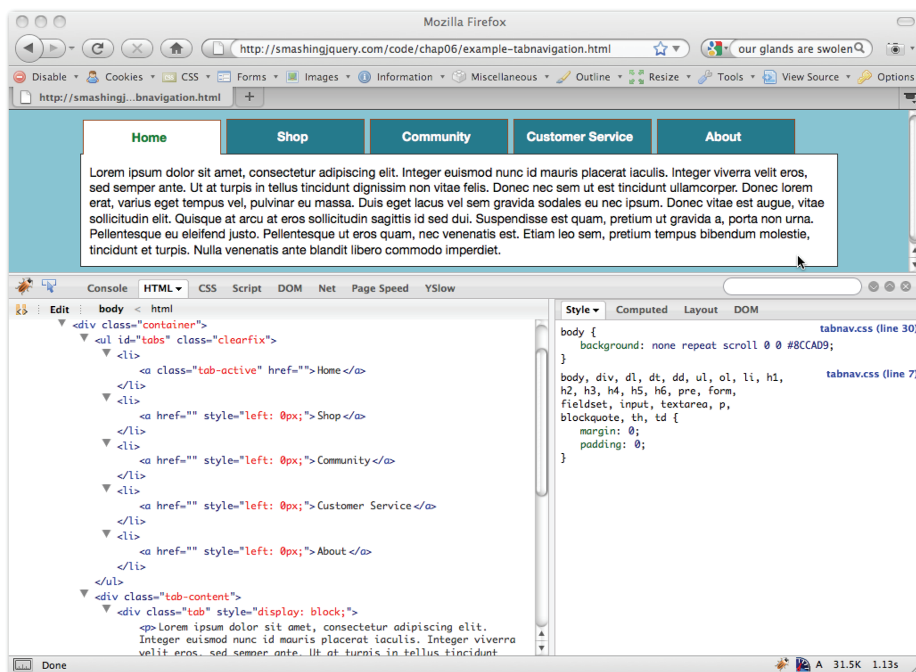


图6-12　Firefox浏览器中通过Firebug插件显示的标签内容脚本

第 7 章

生成可交互的生动表格

20世纪90年代末，伴随着因特网和万维网爆炸性地增长，由于缺少设计和布局工具，人们经常使用表格做页面设计和布局工作，然而这并非表格的原本用途。在使用表格规划页面布局时，为了实现复杂的设计，只能使用层层嵌套的表格。用表格设计和实现页面布局不仅缺少语义，而且会因为层层嵌套的表格使页面臃肿不堪，对查看和维护这些代码的人来说，简直就是恶梦。如今，绝大部分Web设计师和开发者都使用纯HTML和CSS创建具有丰富语义的设计，不再使用表格布局，从而实现了表现层与内容的分离。

HTML表格之所以被滥用为布局工具，一个最主要的原因是由于CSS姗姗来迟而造成布局工具缺失，甚至在CSS规范问世之后的很多年里，浏览器都不能正确地支持CSS。

表格因此（被滥用于布局）臭名昭著，然而如果将表格正确用于显示列表类数据，就能方便地实现既干净又结构良好的格式。在这一章里，我将分享一些使用jQuery增强表格数据显示效果的小技巧。

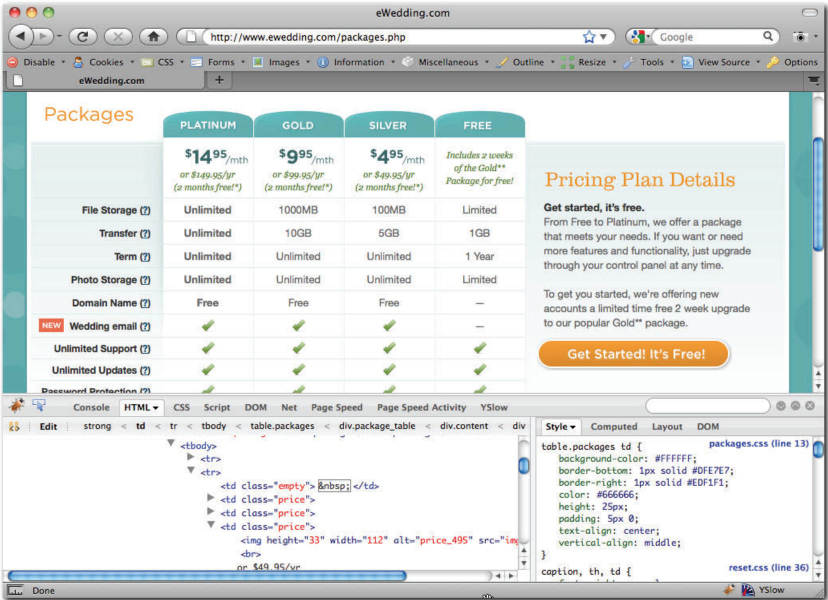
7.1 用 CSS 为表格数据设置样式

图7-1展示的是eWedding网站（www.ewedding.com/packages.php）上一个使用表格显示数据的例子。

我们可以通过为表格添加样式让表格更实用并且更方便用户理解。在本解决方案中，我使用下面的HTML表格：

```
<table border="1" cellpadding="4">
  <thead>
  <tr>
    <th>Category</th>
    <th>Product</th>
    <th>Price</th>
    <th>Status</th>
  </tr>
  </thead>
  <tbody>
  <tr>
    <td>Clothing</td>
    <td>North Face Jacket</td>
    <td>$189.99</td>
```

```
      <td>In-stock</td>
    </tr>
    <tr>
      <td>Shoes</td>
      <td>Nike</td>
      <td>$59.99</td>
      <td>In-stock</td>
    </tr>
    <tr>
      <td>Electronics</td>
      <td>LED TV</td>
      <td>$589.99</td>
      <td>Out of stock</td>
    </tr>
    <tr>
      <td>Sporting Goods</td>
      <td>Ping Golf Clubs</td>
      <td>$159.99</td>
      <td>In-stock</td>
    </tr>
    <tr>
      <td>Clothing</td>
      <td>Sweater</td>
      <td>$19.99</td>
      <td>In-stock</td>
    </tr>
    </tbody>
  </table>
```



由ewedding.cp许可复制

图7-1　一个用表格呈现数据的例子（eWedding网站上的一些数据比较）

7.1.1　使用过滤器创建条纹表格

Web设计师经常使用条纹效果，通过为奇数行和偶数行设置不同的背景色让表格更容易阅读。图7-2就是一个来自Performable网站（www.performable.com）的条纹表格示例。我们可以使用后端编程语言（如PHP或ASP.net）实现条纹效果，但那样的话还需要一位程序员参与，而使用jQuery提供的:even和:odd过滤器（无需劳程序员大驾），能极其容易地为任何表格设置此类样式。

由performable.com许可复制

图7-2　Performable.com 站点上一个使用了条纹效果的表格

在document的ready事件处理函数中添加两条语句，使用jQuery过滤器分别选取表格的奇数行和偶数行。第一条语句选中所有的偶数行并应用浅灰色背景（#dedede）。第二条语句确保所有的奇数行使用白色背景。图7-3展示的是页面加载完成后该例子在浏览器中的输出。

```
$(document).ready(function() {
  $('tbody tr:even').css('background', '#dedede');
  $('tbody tr:odd').css('background', '#ffffff');
});
```

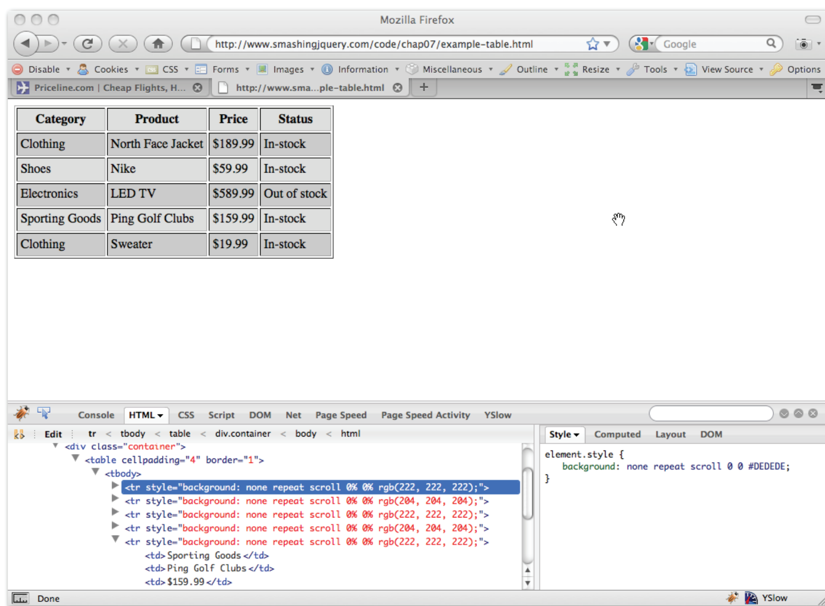


图7-3　页面加载完成后的浏览器输出（所有的偶数行都变成了浅灰色背景）

7.1.2　为表格中的行添加简单悬停效果

我们可以为表格中的行添加悬停效果，让表格对用户的鼠标悬停动作出适当的反应。在下面的例子中，当用户把鼠标放到表格中任意一行时，我们通过为当前行设置另一种背景色轻松地实现了这个效果。图7-4展示的是在Firefox中这个例子的最终效果。

(1) 在document的ready事件处理函数中，为每个tr元素绑定hover事件。这个选择器的好处是它会自动选取表格中所有的行，然后使用寥寥几行代码为它们绑定hover事件处理函数。

```
$(document).ready(function(){
  $('tr').hover(function() {
    }, function() {
  });
});
```

(2) 往hover事件处理函数中添加两条语句，一条加到mouseenter事件中，一条加到mouseleave事件中。我们使用this关键字引用触发事件所在的当前tr元素，在mouseenter事件中将行的背景色设置为粉红色，在mouseleave事件中把行的背景色变回白色。

```
$(document).ready(function(){
  $('tr').hover(function() {
    $(this).css('background','pink');
  }, function() {
    $(this).css('background','white');
  });
});
```

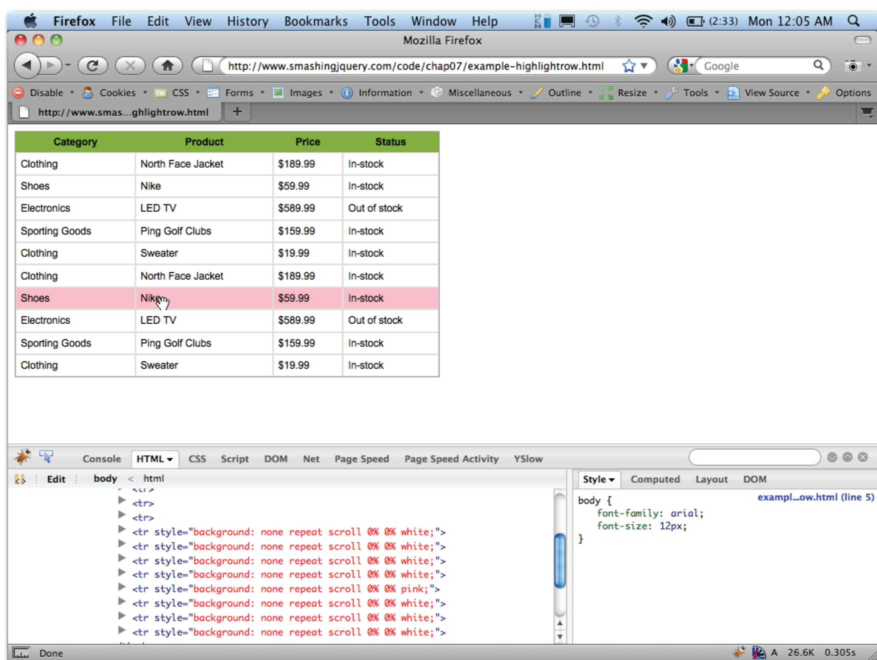


图7-4　当用户将鼠标移动到某一行上时Firefox浏览器中的输出（Firebug已开启；另见彩插图7-4）

7.1.3　为表格中的行添加高级悬停效果

如果我们深入一步，就能做到如图7-5所示，当用户在某行上悬停时显示编辑选项（此例中为Edit Me）。这种用户界面解决方案经常用于适合即时修改数据的场合。

首先我们在document的ready事件处理函数中为所有的tr元素绑定hover事件。添加两条选择器语句，一条用于mouseenter事件，一条用于mouseleave事件。在mouseeneter事件中，我们使用this关键字选中当前元素，接着调用append方法在DOM中插入一个a（链接）标签。当用户在某一行上悬停时，第一条语句为该行的每一列追加 Edit Me（编辑）链接。第二条语句在鼠标离开时将这些追加元素（.editme）移除。

```
$(document).ready(function(){
  $('tr').hover(function() {
    $(this).children().append('<div class="editme"><a href="">Edit Me</a></div>');

  }, function() {
    $('.editme').remove();
  });
});
```

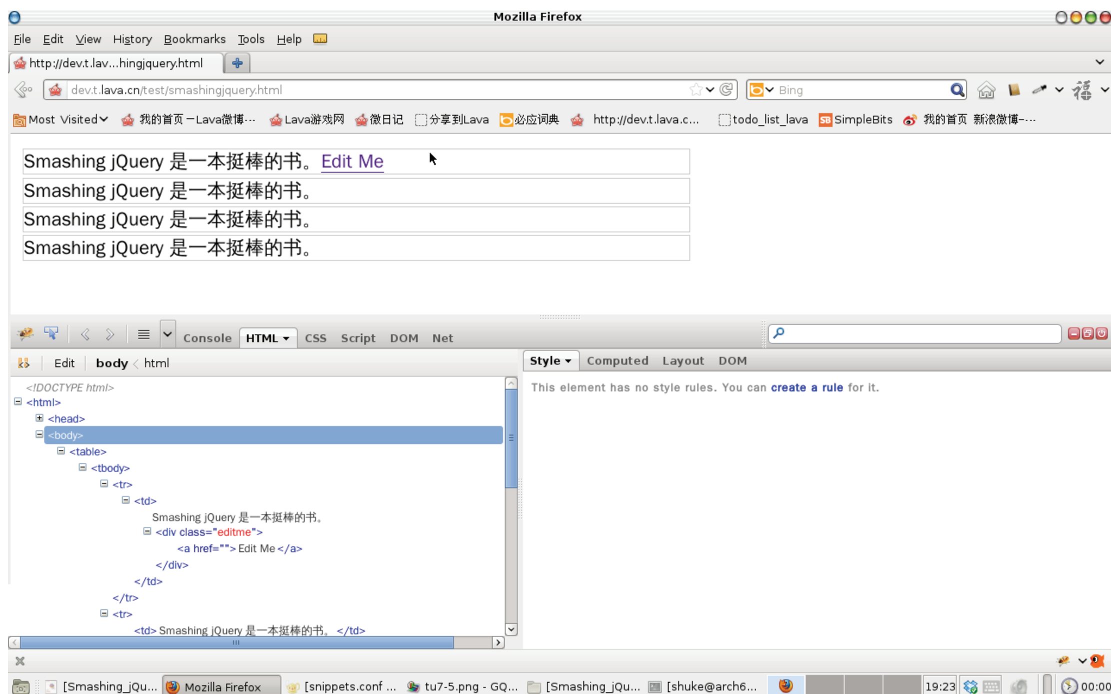


图7-5　鼠标悬停时显示即时编辑选项[①]

7.2　维护表格数据

现在我们已经学会为表格设置简单的样式以及特效，接下来学习维护表格中的数据。这里说的“维护”是指添加、删除和过滤表格数据，这些行为都会改变DOM。前面提到的所有选择器、事件及特效都能用于表格数据。没错，我们可以为所欲为。我会在本节讲解如何完成以下任务：

- 在表格中的第一行或最后一行之后增加一行；
- 基于行号（索引），在具体某一行之后增加一行；
- 在特定内容的行之后增加一行；
- 使用过滤器选择器删除一行；
- 基于行号（索引）删除一行；
- 删除具有特定内容的一行。

在本节的所有教程中我都使用由以下HTML构成的表格。注意它的结构，图7-6是该表在Firefox浏览器中的呈现。

① 原书附图有误，译者根据代码自行制作了图7-5。

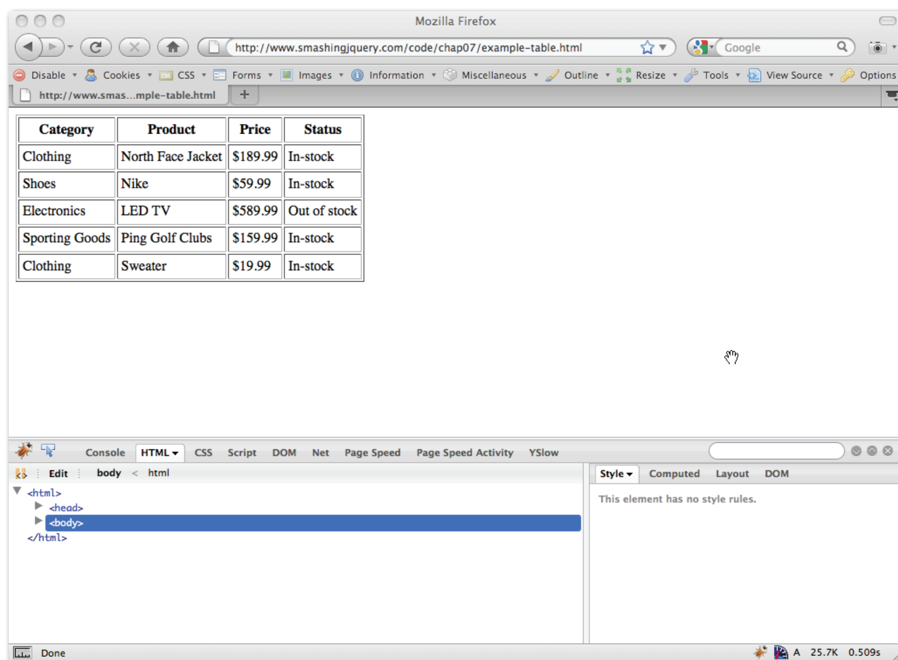


图7-6　示例代码对应的表格在Firefox浏览器中的呈现

```
<table border="1" cellpadding="4" id="products">
  <thead>
  <tr>
    <th>Category</th>
    <th>Product</th>
    <th>Price</th>
    <th>Status</th>
  </tr>
  </thead>
  <tbody>
  <tr>
    <td>Clothing</td>
    <td>North Face Jacket</td>
    <td>$189.99</td>
    <td>In-stock</td>
  </tr>
  <tr>
    <td>Shoes</td>
    <td>Nike</td>
    <td>$59.99</td>
    <td>In-stock</td>
  </tr>
  <tr>
    <td>Electronics</td>
    <td>LED TV</td>
    <td>$589.99</td>
    <td>Out of stock</td>
```

```
    </tr>
    <tr>
      <td>Sporting Goods</td>
      <td>Ping Golf Clubs</td>
      <td>$159.99</td>
      <td>In-stock</td>
    </tr>
    <tr>
      <td>Clothing</td>
      <td>Sweater</td>
      <td>$19.99</td>
      <td>In-stock</td>
    </tr>
    </tbody>
  </table>
```

7.2.1 在表格第一行或最后一行之后添加一行

我们可以利用服务器端程序实现动态添加页面内容，这需要请求服务器，而且根据信息来源数据库的容量，可能需要大量处理时间和很高的处理能力。还有一种更简单的方式，这就是使用jQuery动态插入内容。无论是插入储存在某个jQuery函数中的静态内容，还是通过Ajax载入内容，都不会干扰页面的其他行为。

假设打算在一个搜索结果页的第一条搜索结果之后插入一条特殊信息，我们能够在表格中指定目标位置，以达到这一目的。这有点类似于谷歌总是把付费结果放在每个搜索结果页的最前头，把所有其他结果放到付费结果之后。

jQuery支持我们组合使用`:first`过滤器和`.after()`方法实现这一效果。Priceline网站（www.priceline.com）在旅店搜索结果页采用了类似于谷歌的做法。如图7-7所示，Priceline网站上每隔两家旅店就插入一条特价信息。

接下来我将利用前面示例的HTML表格，演示一下在表格的第一行数据之后插入一条特定信息有多么容易。

(1) 新建一个`special`类，为它设置以下样式，这样在把它添加到表格之后便能够突出显示。

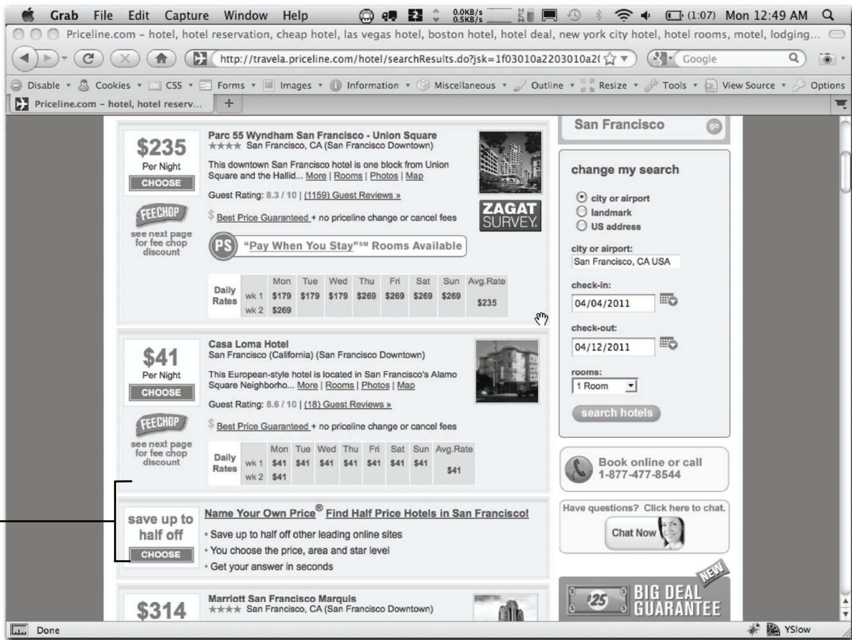
```
.special
 {background:#6AAF18;text-align:center;font-size:22px;color:#fff;font-
  weight:bold;}
```

(2) 在`document`的`ready`事件处理函数中添加一行语句，选中第一个`tr`，然后使用`.after()`方法在该行之后插入今日特价信息。

```
$(document).ready(function() {
  $('#products tr:first').after('<tr><td colspan="4" class="special">Special
  Offer TODAY</td></tr>');
});
```

还有一种可选方案，即使用`:last`过滤器在表格的最后一行之后展示特定内容。图7-8所示为以上脚本在Firefox中的运行结果。

7



每隔两条旅店信息，就插入一行特价信息

© 1998~2010，priceline.com

图7-7　在旅店搜索结果页，Priceline每隔两条旅店信息插入一行特价信息

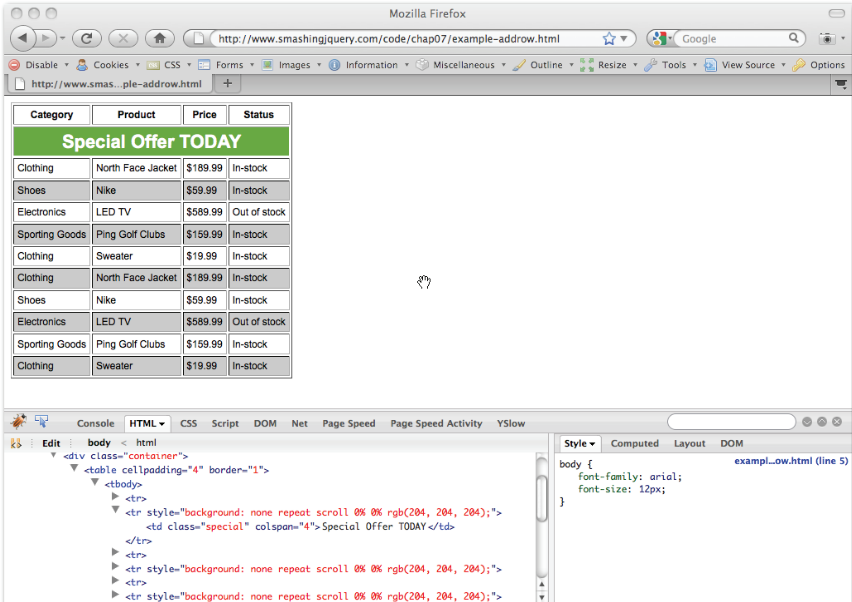


图7-8　Firefox中显示的特价脚本运行结果

有几个扩展特价信息解决方案功能的点子：

- 加一个定时器，在页面呈现10 s之后再显示特价信息，然后等30 s，再让特价信息消失；
- 使用`.animate()`方法为特价信息添加高级动画效果；
- 利用cookie，仅为初次访问的用户显示特价信息。

7.2.2 使用过滤器选择器删除一行

我在工作中经常遇到删除内容的需求，不管是临时删除还是永久删除，都相当不容易。我们每周发布一次代码，代码上线之前需要编译并充分测试。对网站上绝大多数的页面来说，这种每周一次的发布方式无法做到在需要时对页面进行及时修改。如果需要快速修正一个小问题，比方删除某个产品、图片、文本或者DOM中的某类元素，我们可以使用jQuery的`.remove()`方法，由于无需后端程序员掺和这件事，他们很开心。有些时候，在工作时间之外（程序员都已经下班）支持有限的情况下，网站上会出现不该出现的内容。这时能够使用jQuery快速地清理掉这些内容可以及时挽回败局，程序员们也得以在第二天从容地从根上解决这些问题。

`.remove()`方法负责从DOM中删除匹配选择器的元素。

在document的ready事件处理函数中，使用选择器选中元素，然后调用`.remove()`方法。在本例中选择器匹配的是表格的最后一个tr元素，下面这条语句把它从DOM中删除。图7-9展示的是表格最后一行被删除之后的结果。

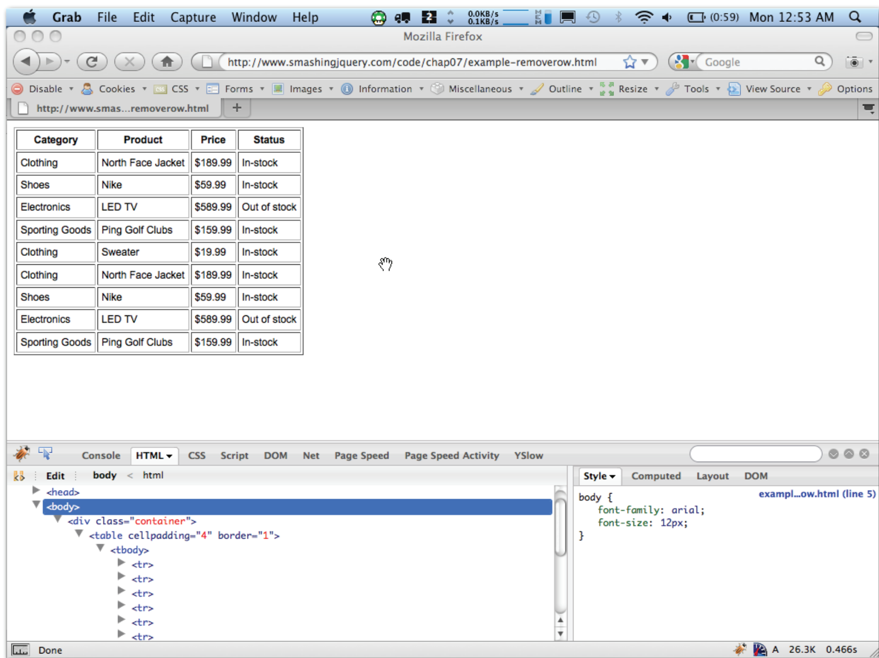


图7-9 表格的最后一行被删除

```
$(document).ready(function(){
  $('tr:last').remove();
});
```

7.2.3　基于索引在某一行之后增加一行

正如同我们基于表格行的位置找出首行和末行，我们也可以基于行号（索引）找出具体某行，然后在它的前面或后面插入一行。在选择器中使用:eq(5)过滤器可以找出索引值为5的所有元素，并在这些元素后面插入Special Offer Today（今日特价信息）。

> :eq()是一个基于索引的过滤器，帮助我们根据索引定位元素。

```
$("tr:eq(5)").after('<tr><td colspan="4">Special Offer TODAY</td></tr>');
```

7.2.4　基于索引删除某行

类似于基于索引添加行，我们也可以基于索引删除行。在DOM中选中索引值为1的所有tr元素，然后调用.remove()删除它们。当然在本例中，实际上只有一行：

```
$("tr:eq(1)").remove();
```

7.2.5　在包含特定内容的行之后追加消息

如果需要在包含特定内容的行之前或之后插入一些行，我们可以用:contains过滤器。类似前面图7-7中Priceline网站的例子，我们可以在表格中使用这个过滤器呈现一条特定的消息给用户看。

考虑这样的场景：我们在表格中找出每一个包含Clothing字符串的行，然后在这些行之后追加一条特定消息，也就是特价信息的HTML代码。这是从DOM中挑出特别的元素并突出显示它们的一种很聪明的方式。注意，:contains过滤器区分大小写。如果目标字符串都是小写的，而我们指定了大写的过滤字符，就不会发生任何匹配。

在页面中使用tr选择器，结合:contains过滤器，我们能找出表格中所有包含字符串Clothing的行，然后在每个匹配行之后插入一条特价信息。图7-10展示的就是使用:contains过滤器添加了特价信息之后的浏览器输出。

> :contains过滤器允许我们基于DOM元素的内容过滤元素。

```
$('tr:contains("Clothing")').after('<tr><td colspan="4" class="special"
  >Special Offer TODAY</td></tr>');
```

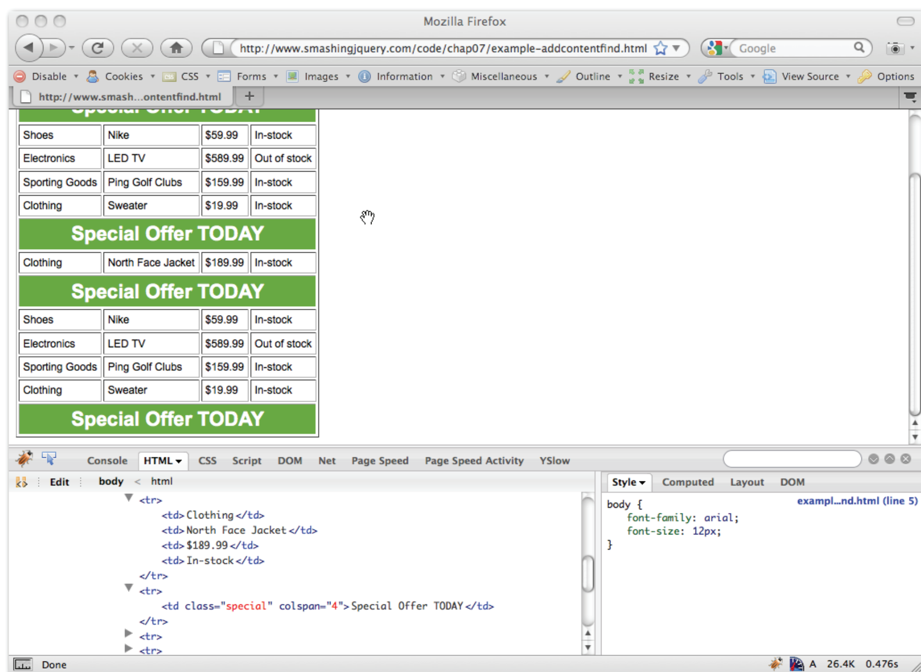


图7-10　使用:contains过滤器添加特价信息

7.2.6　基于元素内容删除一行

如同能基于具体内容添加行，我们也可以基于行内的特定内容删除行。下面这行语句选中含有Clothing字符串的所有tr元素，然后将它们从DOM中移除。

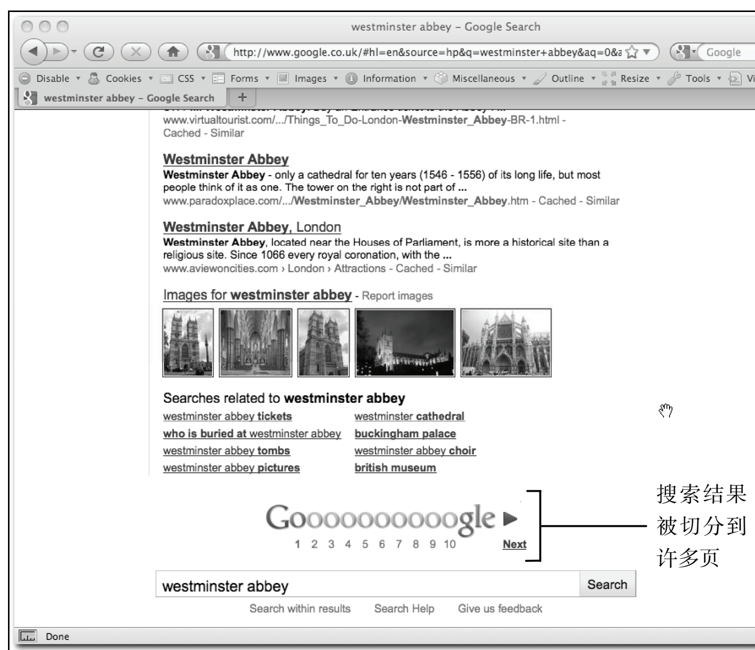
```
$("tr").remove(":contains('Clothing')");
```

7.3　使用 jQuery 设置表格分页

分页就是按照页面大小将内容分成相应大小的片段。在网上，分页无处不在，图7-11展示的是谷歌搜索结果页的分页。利用分页技术我们每次可只呈现少数结果，便于用户导航和理解。

分页经常是由服务器端编程语言（如PHP、ASP.NET和Java）完成。使用服务器端语言解决方案的最大好处是网页不必一次载入全部内容，仅在用户请求时才载入相应的内容页面。我们也可以用jQuery分页，不过我建议把这种技术仅用于总结果较少（不超过100个）的情况，否则会导致页面加载过慢，显著影响网站性能。

如果是老手，你可以利用jQuery做到单击一次只载入10条结果，通过Ajax动态将它们添加到当前页面。



© 2010，Google

图7-11　谷歌搜索结果页上的分页

在下面这个分页例子中，我来讲解如何使用寥寥几行jQuery代码对一个具有很多行数据的表格进行分页。这个脚本自动根据你传入的选择器参数对表格进行分页，并在内容之后生成一个导航菜单，显示可以单击的页码。当单击一个页码时，该页码会被突出显示以便让用户清楚正在显示的是第几页。

(1) 在创建分页脚本之前，我们先准备一些分页需要数据。在本例中我使用表格，但你不必拘泥于此，任何形式的数据（无序列表或一组div）都行。下面的HTML中，表格有12行。

```
<table border="0" cellpadding="0" cellspacing="0" id="data">
  <tr>
    <td>1</td>
    <td>Clothing</td>
    <td>North Face Jacket</td>
    <td>$189.99</td>
    <td>In-stock</td>
  </tr>
  <tr>
    <td>2</td>
    <td>Shoes</td>
    <td>Nike</td>
    <td>$59.99</td>
    <td>In-stock</td>
  </tr>
  <tr>
```

```
    <td>3</td>
    <td>Electronics</td>
    <td>LED TV</td>
    <td>$589.99</td>
    <td>Out of stock</td>
  </tr>
  <tr>
    <td>4</td>
    <td>Sporting Goods</td>
    <td>Ping Golf Clubs</td>
    <td>$159.99</td>
    <td>In-stock</td>
  </tr>
  <tr>
    <td>5</td>
    <td>Clothing</td>
    <td>Sweater</td>
    <td>$19.99</td>
    <td>In-stock</td>
  </tr>
  <tr>
    <td>6</td>
    <td>Clothing</td>
    <td>North Face Jacket</td>
    <td>$189.99</td>
    <td>In-stock</td>
  </tr>
  <tr>
    <td>7</td>
    <td>Shoes</td>
    <td>Nike</td>
    <td>$59.99</td>
    <td>In-stock</td>
  </tr>
  <tr>
  <td>8</td>
    <td>Electronics</td>
    <td>LED TV</td>
    <td>$589.99</td>
    <td>Out of stock</td>
  </tr>
  <tr>
    <td>9</td>
    <td>Sporting Goods</td>
    <td>Ping Golf Clubs</td>
    <td>$159.99</td>
    <td>In-stock</td>
  </tr>
  <tr>
    <td>10</td>
    <td>Shoes</td>
    <td>Nike</td>
    <td>$59.99</td>
    <td>In-stock</td>
  </tr>
  <tr>
```

```
      <td>11</td>
      <td>Electronics</td>
      <td>LED TV</td>
      <td>$589.99</td>
      <td>Out of stock</td>
    </tr>
    <tr>
      <td>12</td>
      <td>Sporting Goods</td>
      <td>Ping Golf Clubs</td>
      <td>$159.99</td>
      <td>In-stock</td>
    </tr>
  </table>
```

(2) 我们利用表格的ID `#data`选取这个表格，然后在它之后插入`div#nav`，使其显示在表格下方。这个`#nav`元素用来存放页码链接。

```
$('#data').after('<div id="nav"></div>');
```

(3) 创建变量`rowsShown`，存放每页展示行数。

```
var rowsShown = 4;
```

(4) 创建变量`rowsTotal`，获取分页数据的总行数，并将它保存于变量`rowsTotal`中。在本例中，先匹配`#data tr`，然后取结果集对象的`length`属性，得到值`12`。

```
var rowsTotal = $('#data tr').length;
```

(5) 创建变量`numPages`，它的值为`rowsTotal`除以`rowsShown`。注意运算结果`Math.ceil`这个JavaScript原生方法进行处理。`Math.ceil`对一个小数进行取舍，得到距离该小数的一个整数。

```
var numPages = Math.ceil(rowsTotal/rowsShown);
```

在所有教程中，创建了一个变量之后，如果想检查它的值，就插入一个`alert`语句并用括号包住这个变量，比如`alert(rowsTotal)`。这个语句不仅与jQuery有关，它是一个原生JavaScript函数，也是最重要的JS调试手段之一。

(6) 接下来创建一个`for`循环，迭代生成用来导航的页码链接。创建变量`pageNum`，它的值为循环变量`i + 1`，这就保证了页码从`1`（而不是从`0`）开始。接下来使用选择器选中`#nav`元素，依次追加页码链接到该元素。页码链接有一个`rel`属性，它的值即链接的序号（从零开始）。链接文本则设置为`pageNum`的值。

```
for(i = 0;i<numPages;i++) {
  var pageNum = i + 1;
  $('#nav').append('<a href="#" rel="'+i+'">'+pageNum+'</a> ');
}
```

(7) 添加一条语句隐藏表格`#data`中所有的行。

```
$('#data tr').hide();
```

(8) 选取表格的第一行并将它显示出来。

```
$('#data tr:first').show();
```

(9) 添加一条语句选取表格#data的所有行，然后使用jQuery的.slice()方法确保只显示前4行。.slice()方法接受两个参数（start和end），以便截取数组的一部分。

```
$('#data tr').slice(0, rowsShown).show();
```

(10) 增加一条语句，选中第一个页码链接，然后为其添加active类。这就保证了页面载入完成后第一个页码处于突出显示状态。

```
$('#nav a:first').addClass('active');
```

(11) 为导航元素#nav内的所有链接绑定click事件处理函数。这个处理函数控制分页，它是这个脚本的核心部分。

```
$('#nav a').bind('click', function(){
});
```

(12) click事件处理函数的第一条语句，移除所有页码链接的active类。

```
$('#nav a').bind('click', function(){
  $('#nav a').removeClass('active');
});
```

(13) 然后为当前被单击的页号链接添加acitve类。

```
$('#nav a').bind('click', function(){
  $('#nav a').removeClass('active');
  $(this).addClass('active');
});
```

(14) 新建变量currPage,保存当前链接的rel属性值(这个值是在前面的for循环中设置的)。

```
$('#nav a').bind('click', function(){
  $('#nav a').removeClass('active');
  $(this).addClass('active');
  var currPage = $(this).attr('rel');
});
```

(15) 再建一个变量startItem，它的值是currPage和rowsShown的乘积。

```
$('#nav a').bind('click', function(){
      $('#nav a').removeClass('active');
      $(this).addClass('active');
      var currPage = $(this).attr('rel');
      var startItem = currPage * rowsShown;
});
```

(16) 再建一个变量endItem，它的值是startItem和rowsShown的和。

```
  $('#nav a').bind('click', function(){
      $('#nav a').removeClass('active');
      $(this).addClass('active');
      var currPage = $(this).attr('rel');
      var startItem = currPage * rowsShown;
      var endItem = startItem + rowsShown;
   });
```

(17) click事件处理函数中的最后一条语句控制单击页码之后显示哪些记录。使用选择器选取了表格#data的所有行。本教程的最终效果见图7-12。

```
$('#nav a').bind('click', function(){
  $('#nav a').removeClass('active');
  $(this).addClass('active');
  var currPage = $(this).attr('rel');
  var startItem = currPage * rowsShown;
  var endItem = startItem + rowsShown;
  $('#data tr').css('opacity','0.0').hide().slice(startItem,
    endItem).css('display','table-row').animate({opacity:1}, 300);
});
```

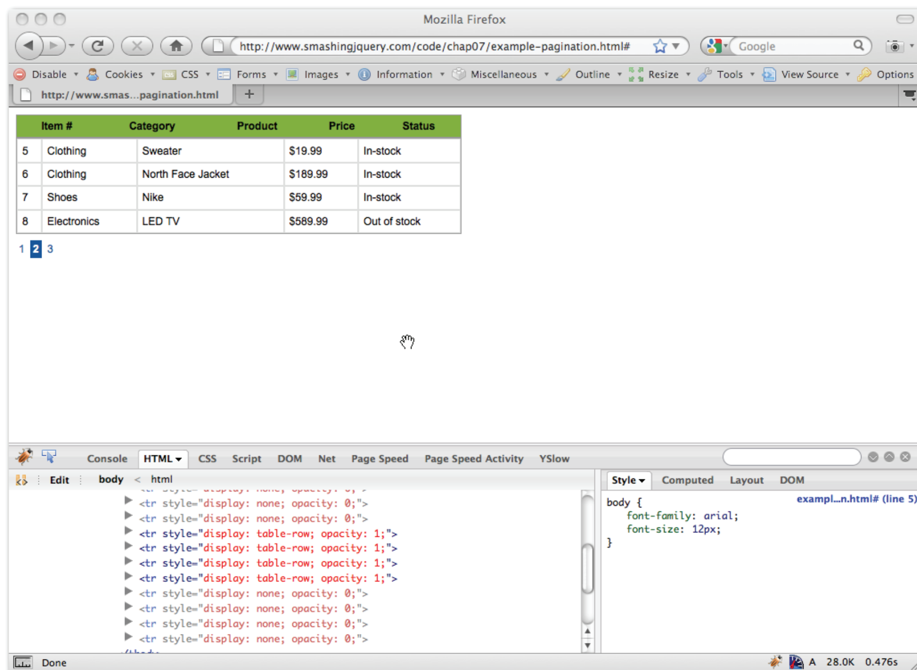


图7-12　分页教程在Firefox及Firebug中的最终输出

7.4　使用 jQuery 插件生成高级表格

jQuery有着非常大的开发者社区，这些人编写了许多jQuery插件。我会在第10章讲解如何编写插件，并点评几个值得推荐的流行插件。jQuery插件唯一的不足之处在于有着太多的选择，有些插件代码质量不高，缺少技术支持。通常我们可以根据随插件提供的文档质量判断一个插件是优是劣。

我们可以利用jQuery自己编写表格排序和过滤功能，不过已经有许多功能强大的插件能做这些事。使用插件能提高解决问题的效率，并且因为绝大多数插件都开源，这为我们提供了一个很好的改进基础。我要在这里特别讲解两个插件——`tablesorter`和`Visualize`。

7.4.1　使用tablesorter插件对表格行排序

tablesorter插件支持我们对任意表格排序。这个插件已经问世好多年了，有着良好的技术支持和极其出色的文档。它支持所有的主流浏览器。表7-1列出了tablesorter方法支持的一些基本配置选项，这些选项使我们能相当灵活地控制排序。下面这个教程使用tablesorter 2.0.5版本。

举个例子，假定你有一个记录数量超过100的表格，希望通过单击某一表头实现针对该列的排序，对tablesorter插件来说，这只是小菜一碟儿。

对表格数据来说，使用jQuery（而不是服务器端排序）的好处在于jQuery排序更快。如果使用服务器端解决方案，每次重新排序都不得不重新载入页面，这就要慢得多。

表7-1　tablesorter 配置选项

选　项	描　述	默　认　值
cssAsc	指定升序排序（由小到大）时应用于标题的类	"headerSortUp"
cssDesc	指定在降序排序（由大到小）时应用于标题的类	"headerSortDown"
cssHeader	指定未排序状态时应用于标题的类	"header"
sortForce	用来指定(在用户选择排序基础上的）强制排序字段	null
sortList	用来指定如何排序的columnIndex和sortDirection参数	null
sortMultiSortKey	指定用于多列排序的键	shiftKey

注：要详细了解tablesorter，请访问 http://tablesorter.com/。

在下面这个例子中，我们一起看看使用tablesorter插件为一个未排序的表格增加排序功能是多么容易。

(1) 在应用tablesorter插件之前，我们需要先准备好表格，否则插件无法工作。我们需要把所有的标题包裹于thead标签中，所有的表格数据单元包裹于tbody标签中，就像下面这个表格的HTML代码一样。

```
<table border="0" cellpadding="0" cellspacing="0" id="data">
  <thead>
    <tr>
    <th>Item #</th>
    <th>Category</th>
    <th>Product</th>
    <th>Price</th>
    <th>Status</th>
    </tr>
  </thead>
  <tbody>
    <tr>
```

```
    <td>1</td>
    <td>Clothing</td>
    <td>North Face Jacket</td>
    <td>$189.99</td>
    <td>In-stock</td>
  </tr>
  <tr>
    <td>2</td>
    <td>Shoes</td>
    <td>Nike</td>
    <td>$59.99</td>
    <td>In-stock</td>
  </tr>
  <tr>
    <td>3</td>
    <td>Electronics</td>
    <td>LED TV</td>
    <td>$589.99</td>
    <td>Out of stock</td>
  </tr>
  <tr>
    <td>4</td>
    <td>Sporting Goods</td>
    <td>Ping Golf Clubs</td>
    <td>$159.99</td>
    <td>In-stock</td>
  </tr>
  <tr>
    <td>5</td>
    <td>Clothing</td>
    <td>Sweater</td>
    <td>$19.99</td>
    <td>In-stock</td>
  </tr>
  <tr>
    <td>6</td>
    <td>Clothing</td>
    <td>North Face Jacket</td>
    <td>$189.99</td>
    <td>In-stock</td>
  </tr>
  <tr>
    <td>7</td>
    <td>Shoes</td>
    <td>Nike</td>
    <td>$59.99</td>
    <td>In-stock</td>
  </tr>
  <tr>
    <td>8</td>
    <td>Electronics</td>
    <td>LED TV</td>
```

```
    <td>$589.99</td>
    <td>Out of stock</td>
  </tr>
  </tbody>
</table>
```

(2) 在使用jQuery插件时，我们总是需要在页头将它包含到页面中。[①]我们必须在使用插件之前先载入插件。没有用到该插件的其他代码可以放在该插件之前且但必须在jQuery库之后（必须哦！）加载。

```
<script src="js/tablesorter.min.js"></script>
```

(3) 在document的ready事件处理函数中，写一条语句选中#data表格，然后调用.tablesorter()方法。如果我们调用tablesorter方法时未提供任何选项，就会自动应用这些选项的默认值（参见表7-1），表格排序效果见图7-13。

```
$(document).ready(function(){
  $("#data").tablesorter();
});
```

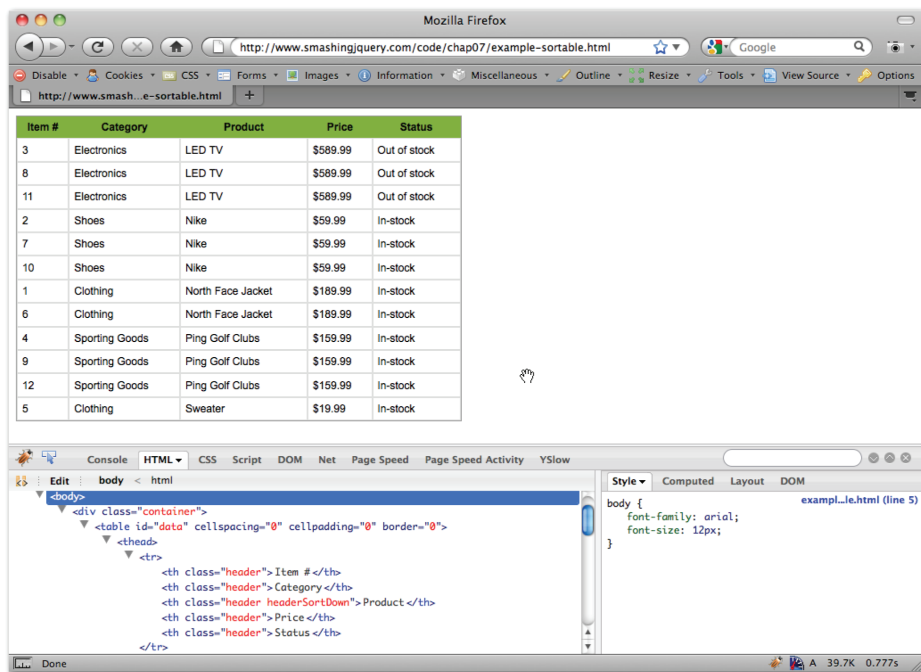


图7-13　tablesorter插件已经应用到前面HTML代码对应的表格上

① 为防止脚本阻塞页面渲染，目前更常见的实现是将所有脚本文件包裹于页面的body结束标签之前。

7.4.2　修改默认排序顺序

通过传入sortList这个数组参数，我们能够改变默认排序的顺序：

```
[columnIndex, sortOrder]
```

在前面我们调用tablesorter方法的那条语句中，这次传入sortList选项参数。数组中的第一个数字是列索引，第二个则指定排序方式（0表示升序，1表示降序）。

```
$("#data").tablesorter({sortList:[[1,0]]});
```

如果希望精确控制tablesorter的行为，可以使用表7-1中的任意一个选项。使用tablesorter插件能非常容易地为任意表格快速添加排序功能。使用tablesorter插件，我们能节省许多编写JavaScript代码的时间，从而把更多的时间用于设计更易使用、更美观的表格。通过传递不同的CSS类选项参数给tablesorter方法，我们可以定制表格的外观。如果一个页面有多个表格，并且希望这些表格的样式各不不同，我们就必须使用这些选项。

7.4.3　使用Visualize插件为表格数据生成迷人的图表

人们经常使用Adobe Flash生成支持人机交互的动态图表，比如图7-14中Google Analytics。最近，一些开发者编写出了不依赖Flash并可与Flash相媲美的jQuery插件。Filament小组开发的Visualize插件对于生成图表，就是一个较好的选择。Visualize插件支持使用表格数据生成图像、线图、柱状图和饼图。它支持很多选项，这些选项可设置和定制任何页面上从HTML表格中抽取数据生成的图表。表7-2列出了Visualize插件的配置选项。

表7-2　Visualize配置选项

选　　项	描　　述	默认行为
type	图表类型	柱状图
width	图表宽度	表格宽度
height	图表高度	表格高度
appendTitle	是否添加标题	是，会添加标题
title	图表的标题	使用caption标签的内容
appendKey	允许在图上添加颜色图例	是，将标题作为颜色标识
colors	自定义颜色	使用默认颜色
textColors	设置文本颜色	N/A
parseDirection	解析数据的方向	默认X轴
pieMargin	设置饼图周围的空白	默认值：20
pieLabelPos	设置饼图标签的位置	默认位置：饼图之内
lineWeight	设置图表边线的粗细	默认值：4
barGroupMargin	设置柱状图各组之间的空白	默认值：10
barMargin	增加柱状图之间的空白	默认值：1

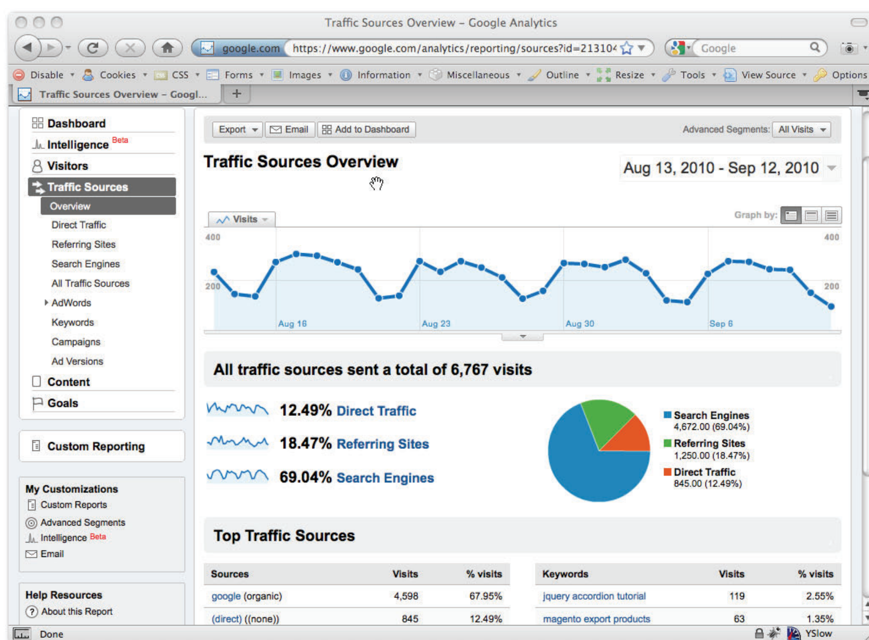


图7-14　Google Analytics使用Flash生成图表

还有一些值得考虑的jQuery图表插件，它们是Highcharts、Flot、jqplot及jQuery Sparklines。

7.4.4　生成柱状图

在下面这个例子中，我们会看到使用Visualize插件基于表格数据生成柱状图有多么容易。最终结果见图7-15。

(1) 在应用Visualize插件之前，我们需要先准备好表格，否则插件无法工作。我们需要把所有的标题包裹于thead标签中，将所有的表格数据单元包裹于tbody标签中，就像下面这个表格的HTML代码一样。我们还要为行和列设置scope属性。

scope属性是一个表格单元（th/td）属性，它用来提示用户（阅读或者使用屏幕阅读器访问页面时）哪些单元是标题。scope属性的可取值为col、colgroup、row和rowgroup，主要用来提高页面的可访问性。

```
<table border="1" cellpadding="4">
  <caption>2010 Traffic</caption>
  <thead>
    <tr>
      <th></th>
      <th scope="col">New Visits</th>
      <th scope="col">Return Visits</th>
    </tr>
  </thead>
  <tbody>
```

```
<tr>
  <td scope="row">Music</td>
  <td>1000</td>
  <td>3500</td>
</tr>
<tr>
  <th scope="row">Sports</th>
  <td>1432</td>
  <td>4633</td>
</tr>
<tr>
到  <th scope="row">Clothing</th>
      <td>1834</td>
      <td>8503</td>
</tr>
<tr>
  <td scope="row">Art</td>
  <td>2543</td>
  <td>3472</td>
</tr>
<tr>
  <td scope="row">Shoes</td>
  <td>4632</td>
  <td>8493</td>
</tr>
</tbody>
</table>
```

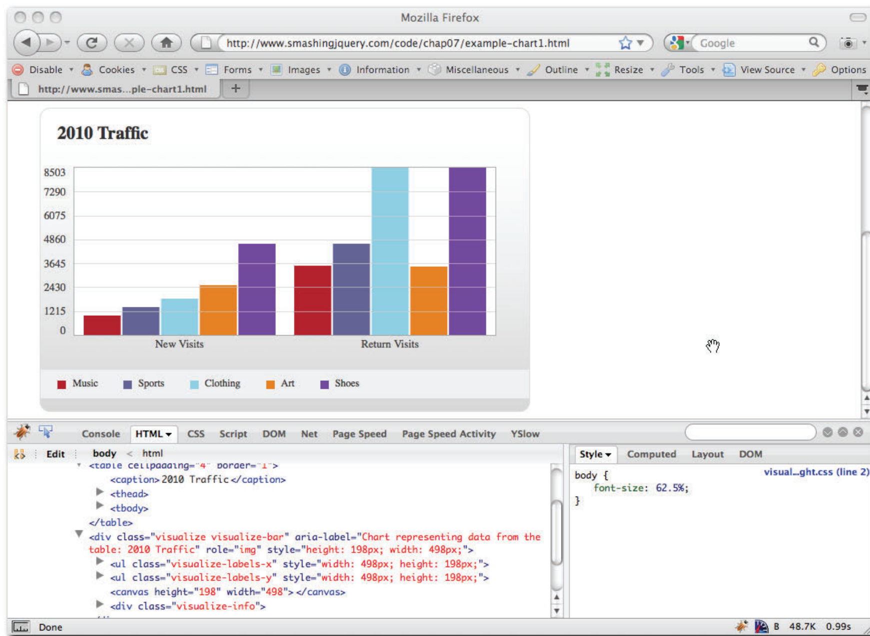


图7-15　使用Visualize插件的表格

(2) 在使用jQuery插件时，我们总是需要在页头将插件包含到页面中。我们必须在使用插件之前先载入插件。没有用到该插件的代码可以放在该插件之前且jQuery库之后加载。

```
<script type="text/javascript" src="js/visualize.jquery.js"></script>
```

(3) 在document的ready事件处理函数中，写一条语句选中#data表格，然后调用.visualize()方法。调用.visualize方法时要传入选项{type:'bar'}。

```
$(document).ready(function(){
  $('table').visualize({type:'bar'});
});
```

第 8 章

使用jQuery制作高级表单

表单在因特网上随处可见，要么是商用表单，要么是注册表单，或者仅仅是一个搜索输入框。表单验证脚本负责保证表单接收到正确的数据，在用户输入无效数据时及时显示清晰的错误信息，并保证提交表单时用户已完整填写了所有必填项。表单验证经常由Web开发者编写的服务器端脚本（使用如PHP、JSP或ASP等编程语言编写）处理。作为服务器端验证（出于数据安全考虑，服务器端验证不可省）的补充，客户端验证越来越常见。会用jQuery和JavaScript的Web开发者与那些（传统上只熟悉）后端的程序员相比能打造更友好的表单用户界面。

我们在第4章讨论事件时提到，jQuery提供了几个专门用于表单的事件（`focus`、`blur`、`change`）。JavaScript早就支持这些表单事件，而jQuery使它们更易于使用。本章，我将和你一起通过几个实例了解在表单处理方面jQuery能帮我们多少。本章中的技术是第9章（讲述了Ajax）的基础。

8.1 页面加载完成后使文本框获得焦点

不管是登录表单还是注册表单，在页面上主动让表单的特定输入项自动得到焦点，能有效地引导用户直接进行下一步操作，这实在是一件方便用户的大好事。如图8-1所示，只需在页面上添加一点点脚本就能使焦点立刻处于正确的输入位置。如果这是一个与转化率有关的商业或零售站点，仅仅这一点点工作就能有效地提高转化率（浏览数到购买数的转化率）。

(1) 准备HTML表单元素。在这个例子中，我们有两个文本输入框，这样页面上就有两个`input`元素，我们接下来演示`:first`过滤器的用法提供了条件。

```
<input type="text1" />
<input type="text2" />
```

(2) 第一条语句选取第一个输入框，然后调用`focus()`方法主动获得焦点。这就保证了页面加载完成之后，第一个文本框自动得到焦点：

```
$('input:first').focus();
```

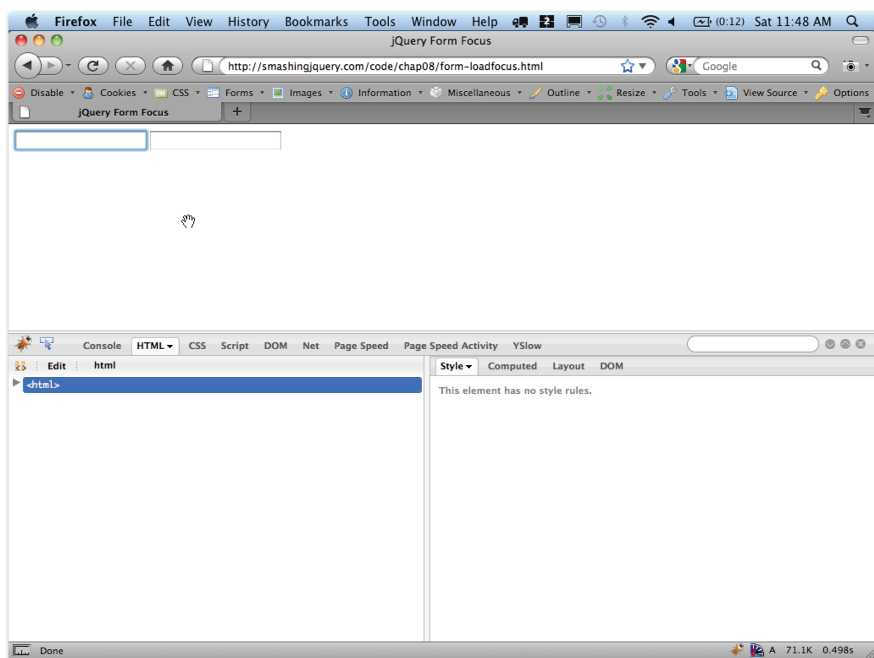


图8-1　Firefox中页面加载完成后，第一个文本框立刻获得了焦点（另见彩插图8-1）

8.2　启用或禁用表单元素

如图8-2所示，有时我们不希望用户修改某个表单项，这时就要启用或禁用表单元素。一种适合禁用表单元素的场合是多页表单。在第一页，我们收集一些类似用户名、密码、电子邮件地址之类的信息。第二页，我们收集账单地址及支付信息，并以禁用表单项的显示方式确认第一页传递过来的信息。

(1) 准备我们希望禁用的HTML文本输入框。

```
<input type="text" id="name-input" />
```

(2) 添加一条语句选中#name-input元素并调用.attr()方法把disabled属性设置为true：

```
$("#name-input").attr("disabled", true);
```

如果希望输入框重新变得可用，只需要将上一条语句中的true参数修改为false。

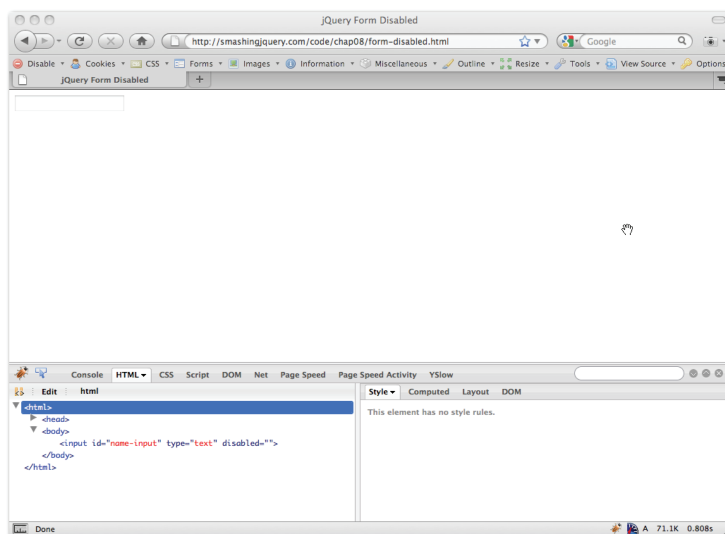


图8-2　调用`.attr()`方法禁用文本框之后的表单

8.3　突出显示表单当前项

在比较大的表单中，我们通过突出显示表单当前项来提醒用户所处位置。绝大多数浏览器都内建了默认突出显示表单当前项的行为，图8-3就是Firefox浏览器中的一个例子。

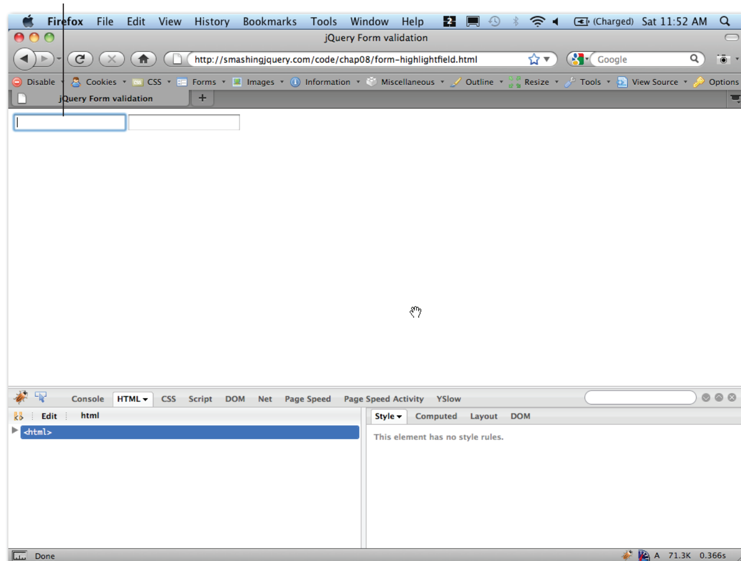


图8-3　Firefox浏览器中内建的表单项突出显示效果（另见彩插图8-3）

我们可以使用CSS和jQuery的focus事件设置辅助的突出显示效果。通过添加自定义的突出显示效果，我们能确保用户知道自己位于表单的什么位置。如图8-4所示，我们可以利用CSS实现类似Wufoo站点（www.wufoo.com）上这样的突出显示效果。

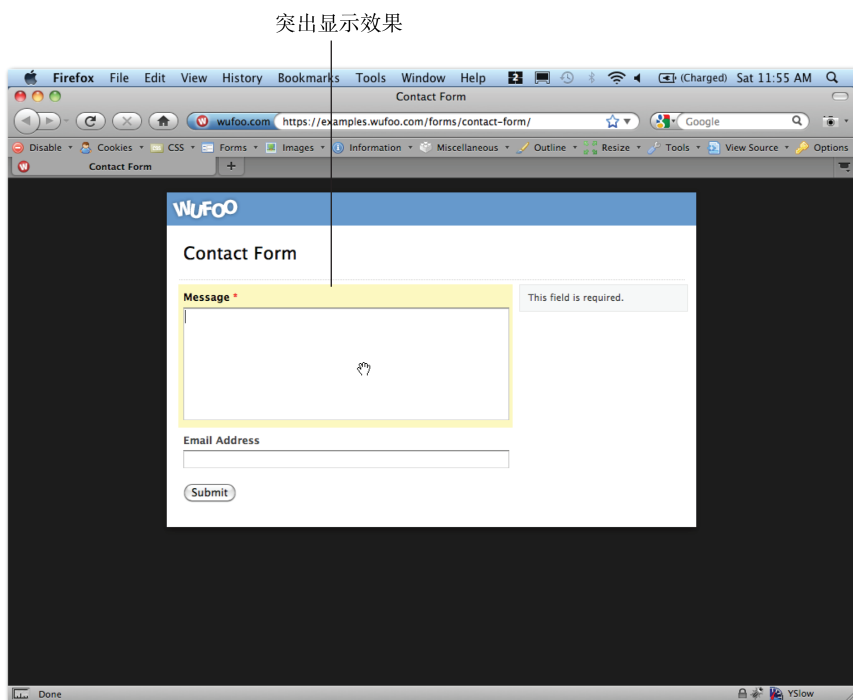


图8-4　Wufoo站点为联系人表单添加了突出显示效果（另见彩插图8-4）

在下面这个例子中，在用户单击鼠标或敲TAB键进入某个表单项时，我们利用focus和blur事件突出显示当前表单项。脚本的最终效果见图8-5。

(1) 准备页面加载完成之后需要突出显示的HTML表单输入元素。

```
<input type="text" id="email-input" />
```

(2) 设置突出显示样式。在本例中，样式为带有5 px填充的黄色背景。

```
.highlight {background:yellow;padding:5px;}
```

(3) 选中所有的input元素并绑定focus事件处理函数。在事件处理函数中添加一条语句，选中this元素，添加highlight类。这样当该input元素得到焦点时（用户单击它或者使用TAB键切换到它时），这个输入项就会自动应用highlight类。

```
$('input').bind('focus', function(){
    $(this).addClass('highlight');
});
```

(4) 选中所有的输入元素并绑定blur事件处理函数。在事件处理函数中添加一条语句，选中this元素，移除highlight类。这样当该input元素失去焦点时（用户离开该表单项时），就会被自动移除highlight类。

```
$('input').bind('blur', function(){
    $(this).removeClass('highlight');
});
```

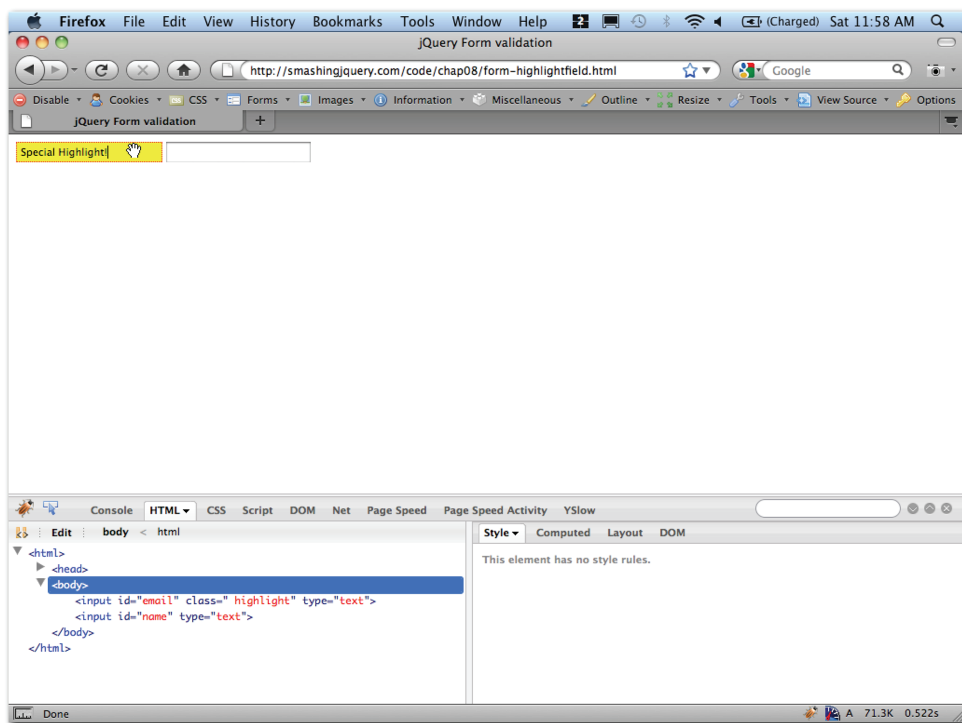
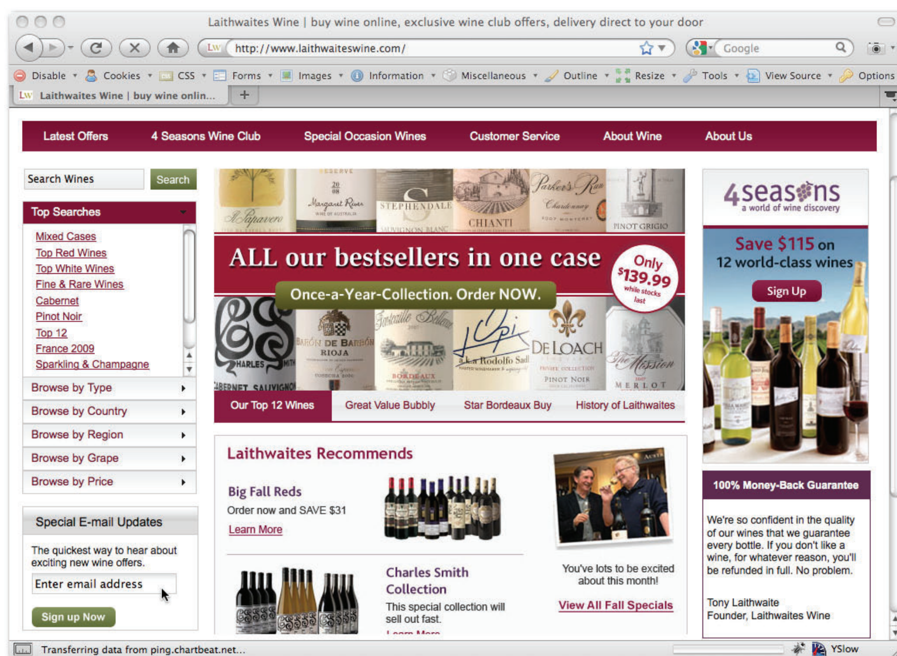


图8-5　Firefox浏览器中突出显示脚本的运行效果（另见彩插图8-5）

8.4　为文本框设置默认文本

为文本框设置默认文本能帮助用户了解这个文本框中应该填写什么内容。为节省空间改进表单设计，许多Web设计师将默认文本用作文本框的标签。我们可以直接使用value属性设置默认值，不过这样有个问题，当用户单击该项时，他们不得不先删除默认值，然后才能输入自己要填写的内容。图8-6展示的是Laithwaites Wine网站（www.laithwaiteswine.com）的一个页面，这个页面上的电子邮件新闻订阅域（位于页面的左下角）就使用了这种简便的默认值设置。



由Laithwaiteswine.com许可复制

图8-6　Laithwaites Wine网站上的一种默认文本的简单实现方式

使用下面的脚本我们可以为一个文本框设置默认文本。如果用户单击该文本框，默认文本消失，然后用户就可以输入自己想填写的内容。如果用户什么也没有填写就离开了文本框，默认文本则会自动重新显示。这个脚本使用focus和blur事件实现了这种效果（工作中的脚本示例见图8-7）。

(1) 准备用来存放默认文本的表单元素input文本框：

```
<input type="text" id="email-input" value="Search"/>
```

(2) 创建一个变量，存放计划显示在文本框中的默认值：

```
var defaultText = "Search";
```

(3) 选中#email-input元素，然后绑定focus事件处理函数：

```
$("#email-input").bind('focus', function(){
});
```

(4) 在focus事件处理函数中，添加一条if语句检查文本框（使用this关键字）的值。如果文本框有值并且它的值与defaultText相同，清除文本框的值。不管文本框是否有值，都把文本的颜色修改为#333（深灰色）。

```
$("#email-input").bind('focus', function(){
  if ($(this).val() == defaultText) {
```

```
    $(this).val('');
  }
  $(this).css('color', '#333')
});
```

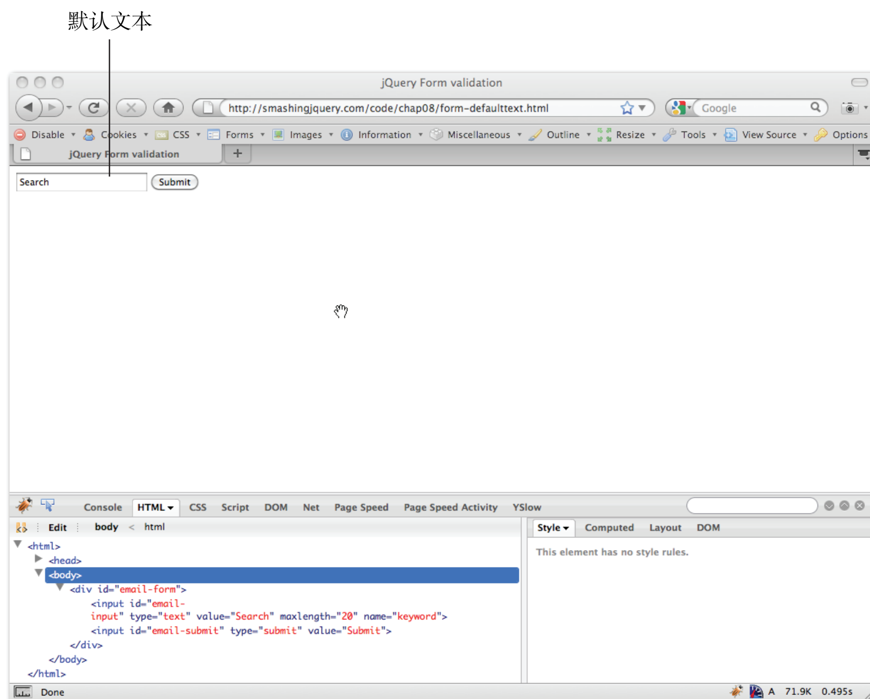


图8-7　页面加载完成之后，文本框中自动填充了默认文本

(5) 接下来，设置#email-input失去焦点时要触发的blur事件。

```
$("#email-input").bind('blur', function(){
});
```

(6) 在blur事件处理函数中，添加一条类似的if语句，检查this的值。如果没有内容，则将它值设置为deafultText。同时，把文本的颜色设置为#FFF。

```
$("#email-input").bind('blur', function(){
  if ($(this).val() == '') {
    $(this).val(defaultText)
  }
    $(this).css('color', '#FFF')
});
```

8.5　限制文本输入框的输入字数

在类似输入个人状态信息等需要限制字数的输入场合，都需要限制字符输入个数的脚本。2010年，Twitter（www. twitter. com）上140个字符的限制使得限制字符个数的脚本的大流行，不过把自己的个人状态信息限制为140个字符或以内终归是一个挑战。当限制用户输入字符数时，字符数限制脚本还能显示出剩余（尚可输入的）字符数。

在本教程中，我将介绍如何利用jQuery `keypress`事件编写如图8-8所示的，类似Twitter上发表短于140字符的状态信息时的提示效果。

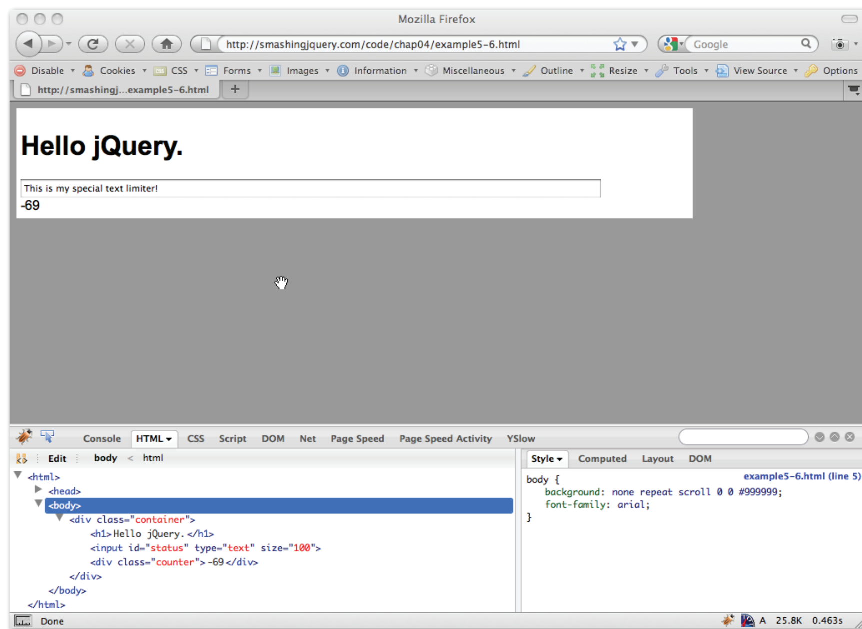


图8-8　在用户输入文本时这个脚本会实时计算剩余字符数

(1) 第一步还是准备HTML。我们使用`textarea#status`。我们将使用它的`keypress`事件检测文本框中输入的字符数。

```
<textarea cols="50" rows="5" id="status"></textarea>
```

(2) 添加一个空白的`div#counter`元素，用它存放输入字符过程中计算出来的剩余字符数。

```
<div id="counter"></div>
```

(3) 新建一个变量`maxNum`，保存允许输入的最大字符数，这里我们把它设置为`100`。

```
var maxNum = 100;
```

(4) 选中`#status`元素，然后为它绑定`keypress`事件处理函数。`keypress`事件发生于一个键被按下然后释放时，它是检测文本框中用户输入的最佳事件。

```
$('#status').bind({
    keypress : function() {
    }
});
```

(5) 当keypress事件发生时，我们需要捕获#status输入框中的内容。我们创建一个变量inputText保存输入框的值，再创建一个变量numChar保存inputText的length属性（长度）。接下来，我们再创建一个变量charRemain，保存剩余字符数。

```
$('#status').bind({
    keypress : function() {
        var inputText = $(this).val();
        var numChar = inputText.length;
        var charRemain = numChar - maxNum;
    }
});
```

(6) 初始化这些变量之后，我们再加一个条件语句检查numChar是否小于等于maxNum。如果小于等于，就选中#counter元素，并修改它的文本为剩余的字符数（charRemain）。

```
$('#status').bind({
    keypress : function() {
        var inputText = $(this).val();
        var numChar = inputText.length;
        var charRemain = numChar - maxNum;
        if (numChar <= maxNum) {
            $('#counter').text(charRemain);
        }
    }
});
```

(7) 最后再加一条else if语句检查numChar是否大于maxNum，如果为真，使用event.preventDefault()阻止用户输入更多的字符。

```
$('#status').bind({
    keypress : function() {
        var inputText = $(this).val();
        var numChar = inputText.length;
        var charRemain = numChar - maxNum;
        if (numChar <= maxNum) {
            $('#counter').text(charRemain);
        } else if (numChar > maxNum) {
            event.preventDefault();
        }
    }
});
```

8.6　实现复选框的全选功能

我们可以用jQuery实现复选框的全选与反选功能，在网站的选项部分经常会用到这一功能。用户会看到20个左右的复选框和一个允许一次单击全部选中的链接。在下面这个教程中，我们使

用jQuery的`click`事件和条件判断实现图8-9所示的Check All（选择全部）复选框。

Check All复选框已经选中

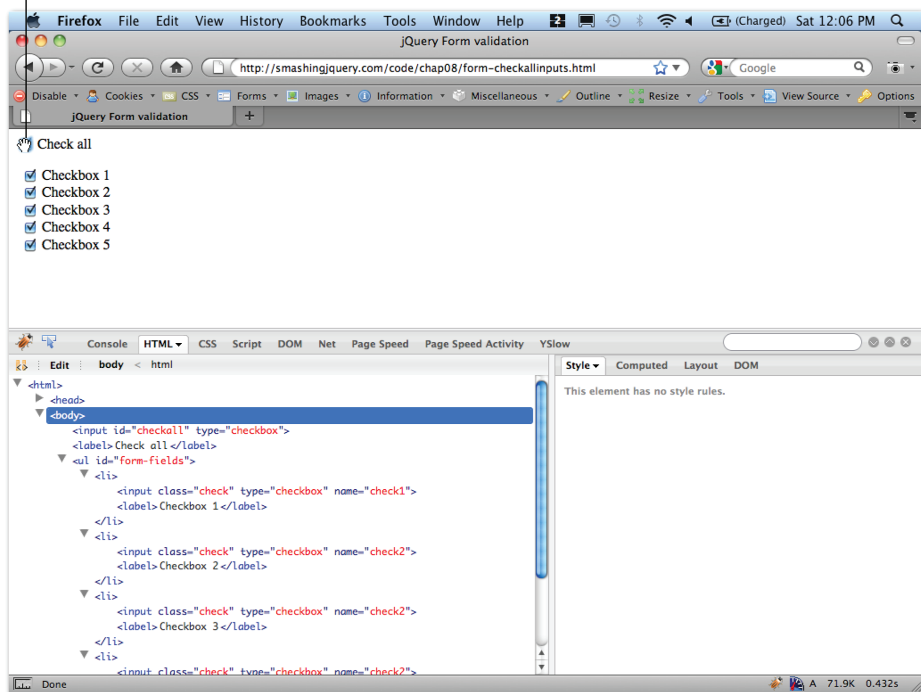


图8-9　Firefox浏览器下的Check All复选框，Firebug面板实时地展示`input`项的变动情况

(1) 首先，我们准备5个复选框元素，将它们置于无序列表`ul#form-fields`中，并为每个复选框元素添加类`check`。

```
<ul id="form-fields">
    <li><input name="check1" class="check" type="checkbox"/> <label>Checkbox
    1</label></li>
    <li><input name="check2" class="check" type="checkbox"/> <label>Checkbox
    2</label></li>
    <li><input name="check2" class="check" type="checkbox"/> <label>Checkbox
    3</label></li>
    <li><input name="check2" class="check" type="checkbox"/> <label>Checkbox
    4</label></li>
    <li><input name="check2" class="check" type="checkbox"/> <label>Checkbox
    5</label></li>
    </ul>
```

(2) 在无序列表之前添加一个复选框`input#checkall`。这个复选框用来控制对下面5个复选框的选中或反选。

```
<input type="checkbox" id="checkall" /> <label>Check all</label>
```

(3) 选中#checkall元素，为它绑定click事件。

```
$('#checkall').bind('click', function(){
});
```

(4) 新建变量checkboxes选中所有的li元素并在其中寻找所有的.check元素（复选框），然后将匹配的结果集保存到变量checkboxes。

```
$('#checkall').bind('click', function(){
  var checkboxes = $('#form-fields li').find('.check'); ①
});
```

(5) 如果我们还需要取消全选功能，可加一条if/else语句检查#checkAll元素的状态，如果是选中状态，就设置保存在checkboxes变量中的复选框结果集中的每一个复选框的checked属性为true，反之则设置为false。

```
$('#checkall').bind('click', function(){
  var checkboxes = $('#form-fields li').find('.check');
  if (this.checked) {
    checkboxes.attr('checked', 'true');
  } else {
      checkboxes.attr('checked', 'false');
  };
});
```

在浏览器中测试上面的代码，你会发现这个管理多复选框选择全部/取消全选的功能既流畅又灵活。

8.7　获取文本输入框的值

如图8-10所示，获取文本框的值是非常基础的表单技术，但这项技术在很多场合都非常有用。它的用法极其容易，可用于许多应用程序。在使用Ajax传送所有表单项的值到服务器时，我经常使用这一技术。我会在第9章和大家一起探讨Ajax应用。

(1) 准备HTML表单输入元素，我们即将获取的就是它的值。

```
<input type="text" name="special" id="my-input" value="Very Cool!"/>
```

(2) 选取#my-input元素并调用.val()方法。

```
$("#my-input").val();
```

为了查看刚刚是否获取到了值，我们需要将#my-input元素的值赋给一个新建变量，然后就可以用alert函数查看是否真的得到了文本框的值。

(3) 新建一个inputVal变量，用它保存第二步得到的值：

```
var inputVal = $("#my-input").val();
```

① 这一句也可以简写为var checkboxes = $('#form-fields input');，更直观，也更容易理解。

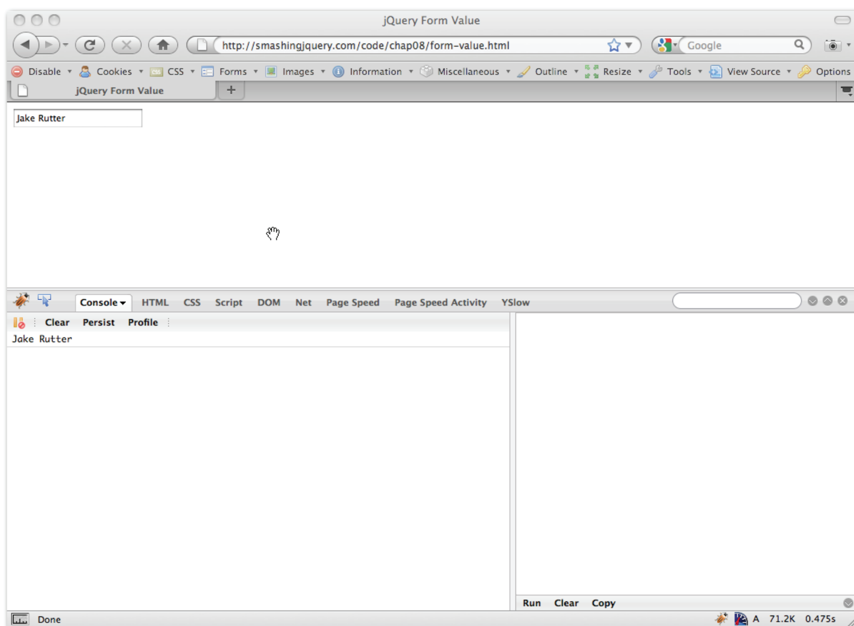


图8-10 通过Firebug控制台捕获并显示出来的值

如果希望在#my-input元素的值改变时得到变化之后的新值，就把上面的语句放到该元素的change事件处理函数中。

(4) 将上面的语句放到#my-input元素的change事件中，这样就保证了输入框中的文字改变时，我们得到的值总是最新的。

```
$('#my-input').change(function() {
  var inputVal = $('#my-input').val();
  alert(inputVal);
});
```

注意，有些浏览器（包括Opera）中，在文本框中的文字发生变化时，change事件并不是马上发生，而是在焦点离开文本框时才发生。

8.8 得到 select 元素的值

在各种站点应用中，使用jQuery获取select元素的值既容易又有用。图8-11展示的是Crutchfiled.com网站（一家汽车／家用音响电器网站）中outfitmycar（配置我的爱车）功能区域使用这个功能的一个例子。Crutchfield允许你选择车的购买年份、型号和制造商。每做一个选择，系统就获取你选择的值，并利用它过滤显示另一个选项列表。

8

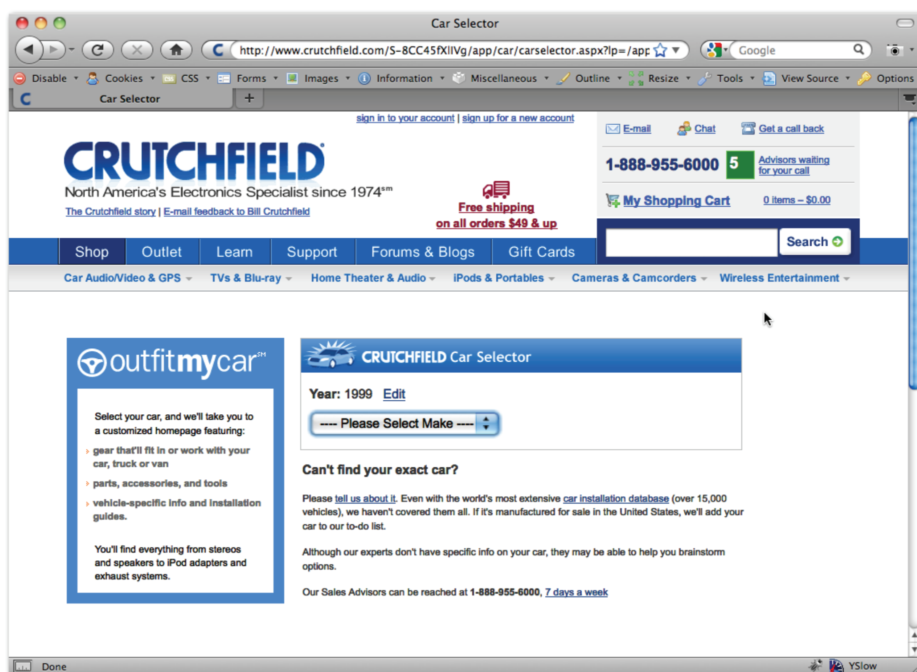


图8-11　Crutchfield.com配置爱车向导

(1) 准备用来获取选项值的select元素。

```
<select id="my-select" name="question1">
  <option value="yes">yes</option>
  <option value="no">no</option>
</select>
```

(2) 选中#my-select元素，然后为它绑定change事件处理函数：

```
$('#my-select').change(function() {
});
```

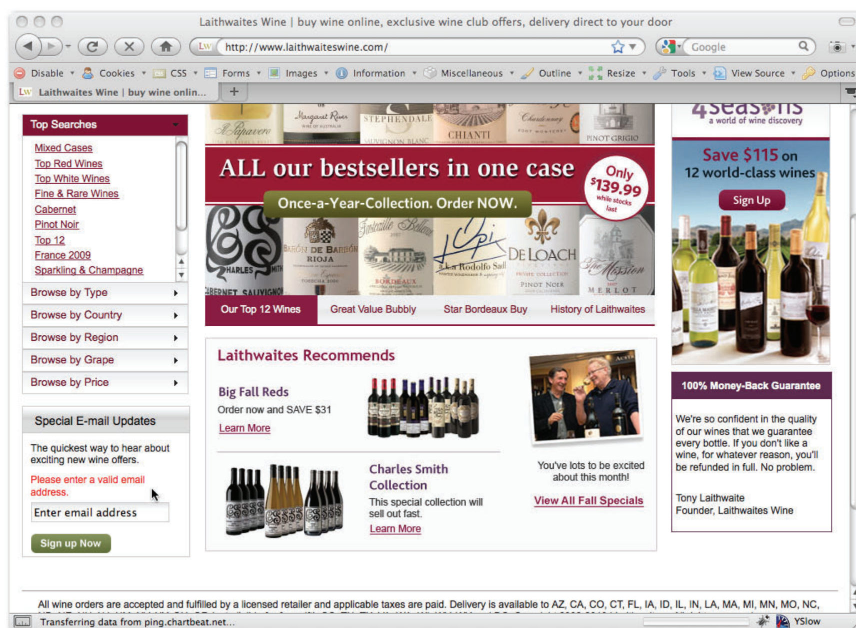
(3) 获取当前选中项的值，并把它赋值给变量selectVal：

```
$('#my-select').change(function() {
  var selectVal = $(this).val();
  alert(selectVal);
});
```

8.9　简单验证表单中的电子邮件

在一个网站上，对消息订阅的注册表单来说电子邮件验证分外重要。如果用户提交了不良的电子邮件地址，数据库清理工作既费时间又费金钱。只需要很少的代码，我们就能使用jQuery为

表单添加简单的电子邮件验证。这类脚本非常有用，就像图8-12所示的注册电子邮件获取特价Laithwaites Wine供应信息的例子，我们并不需要像高级验证程序那样验证好多字段分属不同类型的数据。我会在本章的末尾演示使用一个常见的jQuery验证插件对表单进行高级验证。我们要牢牢记住，永远不要仅仅依赖客户端JavaScript验证，不管客户端验证是否可用，服务器端验证必不可少。



由Laithwaiteswine.com许可复制

图8-12　Laithwaites Wine站点的新消息账号注册链接使用简单的电子邮件验证

下面的代码使用正则表达式检查用户提交的电子邮件地址格式是否正确。`click`事件处理函数绑定在Submit按钮上，因此仅在用户提交表单时才验证。脚本会根据验证结果（是否已输入电子邮件地址及其是否有效）显示成功或各种错误消息。如图8-13所示，当检测到用户输入不符合电子邮件地址格式时，会突出显示一条提示信息，提醒用户应该输入正确格式的电子邮件地址。

正则表达式是匹配文本和数字字符串的特殊模式，广泛用于匹配电子邮件地址、电话号码、邮政编码、信用卡号码等。在绝大多数Web编程语言中，使用正则表达式都是标准做法，我们可以从网上寻找需要的正则表达式用于自己的脚本。

(1) 首先我们创建一个非常简单的电子邮件表单，它只有一个电子邮件输入框和一个Submit（提交）按钮。

```
<div id="email-form">
  <input type="text" id="email-input" name="email"/>
  <input type="submit" value="Submit" id="email-submit" name="submit"/>
</div>
```

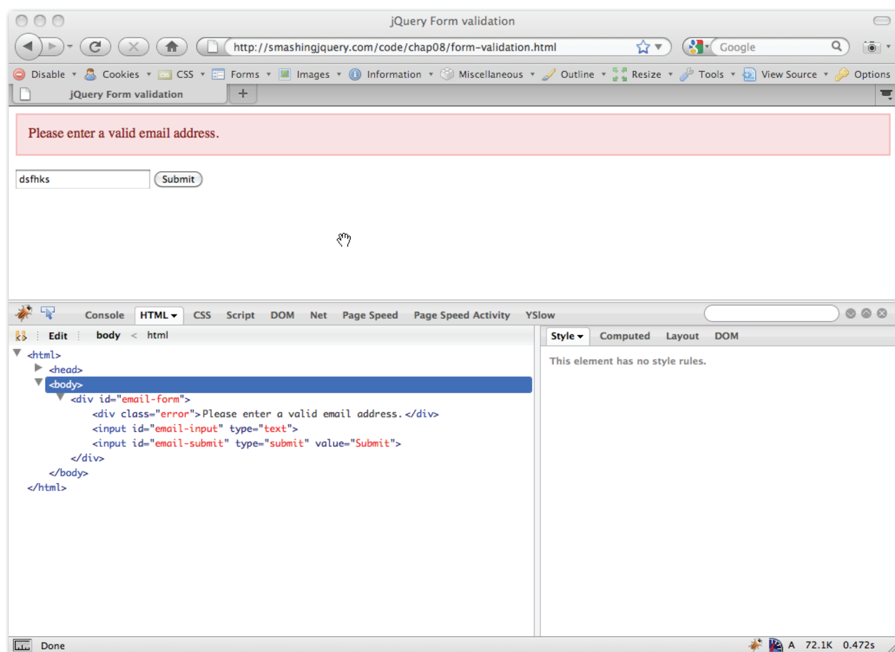


图8-13　若用户输入的电子邮件地址格式不正确，在单击Submit（提交）按钮时会显示一条提示信息

(2) 选中#email-submit按钮并为它绑定click事件处理函数。在click事件处理函数中，添加一条return false语句，这就确保了用户在单击Submit按钮时，脚本会中止表单默认的提交行为。

```
$("#email-submit").bind('click', function(){
  return false;
});
```

(3) 创建一个变量emailReg，然后把检测电子邮件合法性的正则表达式赋值给这个新建变量：

```
$("#email-submit").bind('click', function(){
  var emailReg = /^([a-zA-Z0-9_.-])+@(([a-zA-Z0-9-])+.)+([a-zA-Z0-9]{2,4})+$/;
  return false;
});
```

(4) 创建一个变量email，把#email-input的值赋给它：

```
$("#email-submit").bind('click', function(){
  var emailReg = /^([a-zA-Z0-9_.-])+@(([a-zA-Z0-9-])+.)+([a-zA-Z0-9]{2,4})+$/;
  var email = $("#email-input").val();
  return false;
});
```

(5) 选中#email-form元素并在表单内容之前插入一个div.error。我们使用它存放必要时显示给用户看的出错信息。

```
$("#email-submit").bind('click', function(){
  var emailReg = /^([a-zA-Z0-9_.-])+@(([a-zA-Z0-9-])+.)+([a-zA-Z0-9]{2,4})+$/;
  var email = $("#email-input").val();
  $('#email-form').prepend('<div class="error"></div>');
  return false;
});
```

(6) 添加一条`else if`语句，当用户单击Submit按钮时用它检查用户是否键入了内容。

```
$("#email-submit").bind('click', function(){
  var emailReg = /^([a-zA-Z0-9_.-])+@(([a-zA-Z0-9-])+.)+([a-zA-Z0-9]{2,4})+$/;
  var email = $("#email-input").val();
  $('#email-form').prepend('<div class="error"></div>');
  if(email == '') {
  } else if {
  } else {
  }
    return false;
});
```

(7) 在`if`子句中（用户没有输入任何东西时），选取`div.error`，替换成失败提示信息。

```
$("#email-submit").bind('click', function(){
  var emailReg = /^([a-zA-Z0-9_.-])+@(([a-zA-Z0-9-])+.)+([a-zA-Z0-9]{2,4})+$/;
  var email = $("#email-input").val();
  $('#email-form').prepend('<div class="error"></div>');
  if(email == '') {
    $(".error").replaceWith('<div class="error">You forgot to enter an email
    address.</div>');
  } else if {
  } else {
    }
    return false;
});
```

(8) 在`else if`子句中（用户输入了不合法的电子邮件地址），选中`div.error`元素，替换成错误格式信息。这里的`else if`语句使用正则表达式测试用户输入的电子邮件格式是否合法。

```
$("#email-submit").bind('click', function(){
  var emailReg = /^([\w-\.]+@([\w-]+\.)+[\w-]{2,4})?$/;
          var email = $("#email-input").val();
  $('#email-form').prepend('<div class="error"></div>');
  if(email == '') {
    $(".error").replaceWith('<div class="error">You forgot to enter an email
    address.</div>');
  } else if(!emailReg.test(email)) {
   $(".error").replaceWith('<div class="error">Please enter a valid email
    address.</div>');
  } else {
    }
  return false;
});
```

(9) 最后，添加一条`else`语句，将`#email-form`表单元素的内容替换为一条提示信息，告诉用户他们已经成功订阅。这条消息当且仅当`if`及`else if`语句均检测失败时才显示出来。

```
$("#email-submit").bind('click', function(){
  var emailReg = /^([\w-\.]+@([\w-]+\.)+[\w-]{2,4})?$/;
  var email = $("#email-input").val();
  $('#email-form').prepend('<div class="error"></div>');
  if(email == '') {
    $(".error").replaceWith('<div class="error">You forgot to enter an email
    address.</div>');
  } else if(!emailReg.test(email)) {
    $(".error").replaceWith('<div class="error">Please enter a valid email
    address.</div>');
  } else {
     $("#email-form").html('<div class="success">Thank you, you have been
     subscribed.</div>');
    }
  return false;
});
```

下面是改进这个脚本的一些建议。

- 如果希望在用户输入数据时实现实时检测，可以把绑定到Submit按钮上的`click`事件改为绑定到`input`元素上的`change`事件。这样的话就需要再加一些代码，以保证用户看到正确的提示信息（对输入内容的要求等）。
- 若用户提交了错误格式的电子邮件地址，为电子邮件输入框增加突出显示效果。
- 参考表8-1中常用的正则表达式为表单添加更多的字段，验证其他类型的数据。

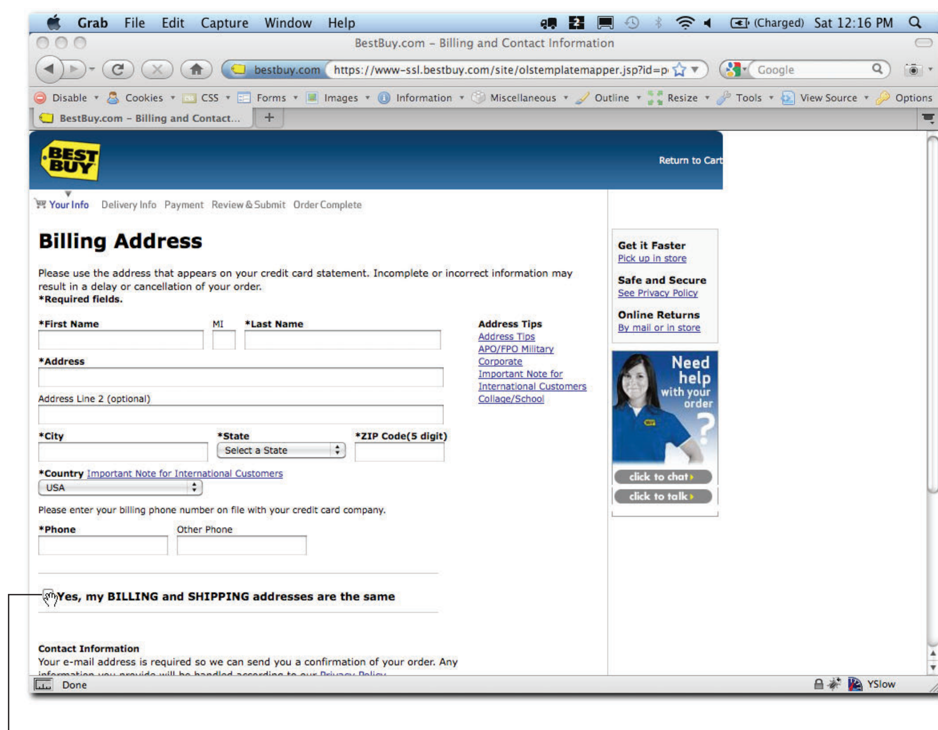
> 正则表达式在网上随处可见，可查阅表8-1了解目前特别流行的正则表达式。

表8-1　另外一些用于表单验证的常见正则表达式

用　途	例　子
电话号码	`(/^[0-9-+]+$/)`
日期（*dd/mm/yyyy*）	`(/^\d{1,2}\/\d{1,2}\/\d{4}$/)`
仅数字	`(^[0-9]+$)`
仅字母	`(^[A-Za-z]+$)`

8.10　复制一个文本框的内容到另一个文本框

如果曾经在网上买过东西，那你很可能有过不得不填写账单地址和收货地址的经历。对绝大多数顾客来说，这两个地址相同，因此这类表单往往有一个Same As Billing Address（与账单地址相同）的复选框，当用户单击该复选框时，脚本就自动复制账单地址为收货地址，从而避免顾客重填一遍。这个复选框使用了复制表单文本框功能。这不仅节省了顾客的时间，也减少了键入错误。图8-14展示的是来自Best Buy站点（www.bestbuy.com）的一个复制功能复选框。



复制功能复选框

图8-14　Best Buy站点上一个复制功能复选框的例子

下面的脚本展示了如何实现复制功能复选框，你可以把它用于任何表单，不一定非要像下面这个例子一样用于账单/收货地址表单。图8-15展示的是该脚本的工作情况。

(1) 首先创建两组地址输入框。第一组用于输入账单地址，第二组用于输入收货地址。在这两组文本框之前添加一个复选框，它负责控制复制账单地址到收货地址这一功能。

```
<label>Copy Fields</label>
<input type="checkbox" id="copy-fields"/>

<div id="billing-address">
<h2>Billing Address</h2>
<label>First Name</label>
<input type="text" id="b-first-name"/>

<label>Last Name</label>
<input type="text" id="b-last-name"/>
</div>

<div id="shipping-address">
<h2>Shipping Address</h2>
<label>First Name</label>
```

```
<input type="text" id="s-first-name"/>

<label>First Name</label>
<input type="text" id="s-last-name"/>
</div>
```

复制功能复选框

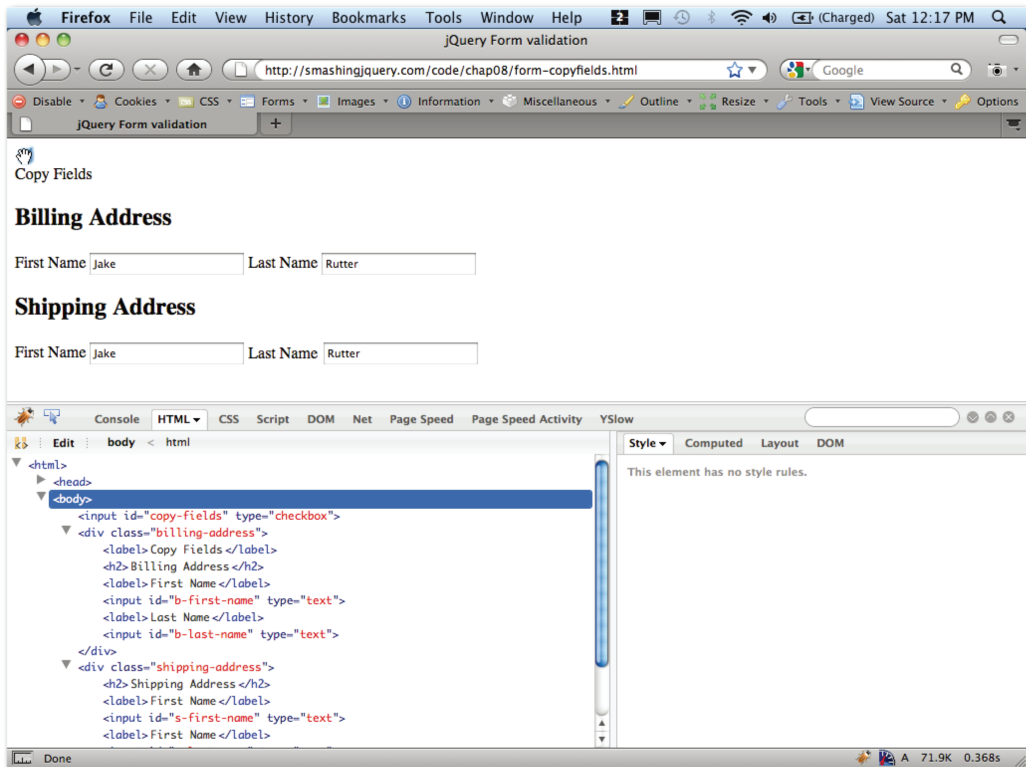


图8-15　工作中的复制功能复选框

(2) 选取#copy-fields元素并为它绑定click事件处理函数。

```
$('#copy-fields').bind('click', function(){
});
```

(3) 为数据复制源建立两个新变量，分别保存账单地址中两个文本框的值：

```
$('#copy-fields').bind('click', function(){
  var billFName = $('#b-first-name').val();
  var billLName = $('#b-last-name').val();
});
```

(4) 在click事件处理函数做任何动作之前，先使用if else语句判断复制功能复选框是否被选中：

```
$('#copy-fields').bind('click', function(){
  var billFName = $('#b-first-name').val();
  var billLName = $('#b-last-name').val();
  if (this.checked) {
  } else {
  };
});
```

(5) 如果复选框为选中状态，将账单地址中相应字段的值复制到收货地址对应的字段：

```
$('#copy-fields').bind('click', function(){
  var billFName = $('#b-first-name').val();
  var billLName = $('#b-last-name').val();
  if (this.checked) {
    $('#s-first-name').val(billFName);
    $('#s-last-name').val(billLName);
  } else {
  };
});
```

(6) 我们还需要在else子句中设置一个备用选项，当用户取消选中复制功能复选框时清除收货地址的内容：

```
$('#copy-fields').bind('click', function(){
  var billFName = $('#b-first-name').val();
  var billLName = $('#b-last-name').val();
  if (this.checked) {
    $('#s-first-name').val(billFName);
    $('#s-last-name').val(billLName);
  } else {
    $('.shipping-address input').val('');
  };
});
```


8.11　利用插件增强表单功能

整合第三方开源插件到你的站点或应用程序，能有效提高开发速度并提供更好的表单验证功能，和更友好的人机交互。由于jQuery有着海量粉丝和支持者构成的巨大的开发者社区，jQuery拥有数不清的支持各种功能的插件，包括设定默认文本的插件、验证表单任意字段的插件等。

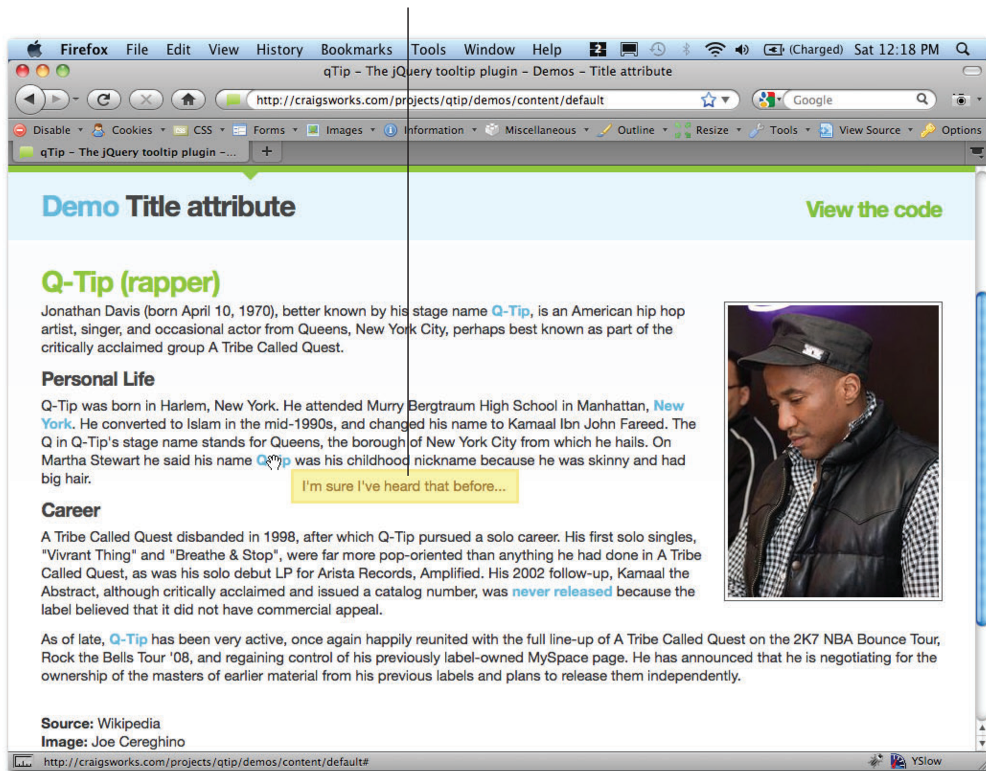
在开发时我经常使用插件，通常是因为时间紧迫无法自己开发某个功能。绝大多数情况，如果能找到某个人已经写好的插件，我就不必自己写了。在本节中，我将介绍两个我经常使用的表单插件——qTip和Validate。

8.11.1　为网站整合qTip插件

qTip是一个jQuery插件，我们可通过它轻松地为任意元素设置高级提示信息。这些提示信息可以是静态内容，也可以是动态内容。qTip支持许多功能，比如多浏览器支持（IE／

Firefox/Safari/Opera/Chrome），这些功能使它成为了一个魅力四射的轻量级插件。对于那些无法完美支持的浏览器，提示信息窗会自动优雅降级。qTip可用于页面的任意元素，包括段落标签（如图8-16所示）。

用qTip创建的一个提示信息



由CraigWorks.com许可复制

图8-16　在qTip站点上，一个段落使用qTip显示提示信息

我喜欢结合表单使用提示信息，提醒用户应该键入什么内容。就像我在第5章中介绍过的那样，我们能够非常容易地从零开始，为某个元素创建基本提示信息。然而qTip拥有完整的API，它支持改变提示信息的样式（以便与我们的站点风格保持一致）、出现位置、动画效果，还有添加动态内容。

使用qTip的最大好处是它支持非常多的配置选项。另外，它的文档也是一流的。这就是我一直用它的原因。有些插件文档极其糟糕，也缺少必要的支持，这使得它们难以使用。偶尔遇到文档很差的插件，我通常都会自己再写一个类似的。

表8-2概要的描述了jQuery qTip插件的一些选项。

表8-2　qTip 插件选项

选　项　名	描　　述
`content`	指定提示信息的内容
`position`	指定提示信息在DOM中的显示位置
`show`	指定显示提示信息的特效
`hide`	指定隐藏提示信息的特效
`style`	指定提示信息的样式
`api`	指定回调函数

注：要查看qTip的完整选项列表，请访问 http://craigsworks.com/projects/qtip/docs/reference/。

8.11.2　利用qTip使用**`title`**属性创建表单元素的基本提示信息

本节，我将演示如何在站点中整合qTip插件并为一个表单元素设置提示信息。这个提示信息指导用户在当前项应该填写的内容。我们将使用`title`属性存放需要显示的文本。使用`title`属性的好处是，在用户禁用JavaScript的情况下，把鼠标放到输入框上时，尽管显示的没有那么美观，仍然可以看到提示信息。

(1) 准备一个文本框，并把它的ID设置为`email`：

```
<input type="text" id="email" />
```

(2) 使用jQuery插件时，我们总是要在页头把这个插件包含到页面中。它的包含位置应该总是刚好位于jQuery库之后，且位于你自己编写的任何引用该插件的jQuery代码之前。

```
<script type="text/javascript" src="js/jquery.qtip-1.0.0-rc3.min.js"></script>
```

(3) 为这个输入框添加一个`title`属性。`title`属性包含的文本将用作提示信息。

```
<input type="text" id="email" title="Please enter your email address."/>
```

(4) 选中所有拥有`title`属性的`input`元素，调用`.qtip()`方法。如果调用`.qtip()`方法时未提供任何选项，`.qtip()`将使用默认设置 。

```
$('input[title]').qtip({});
```

(5) 通过传递选项参数给`.qtip()`方法，我们可自定义qTip。在下面的代码中我为图8-17所示的例子使用了以下选项：

```
$('input[title]').qtip({
    style: { color: 'black', name: 'blue', tip: true },
    position: { corner: { target: 'bottomMiddle' }},
    show: { when: { event: 'focus' }},
    hide: { when: { event: 'blur' }}
});
```

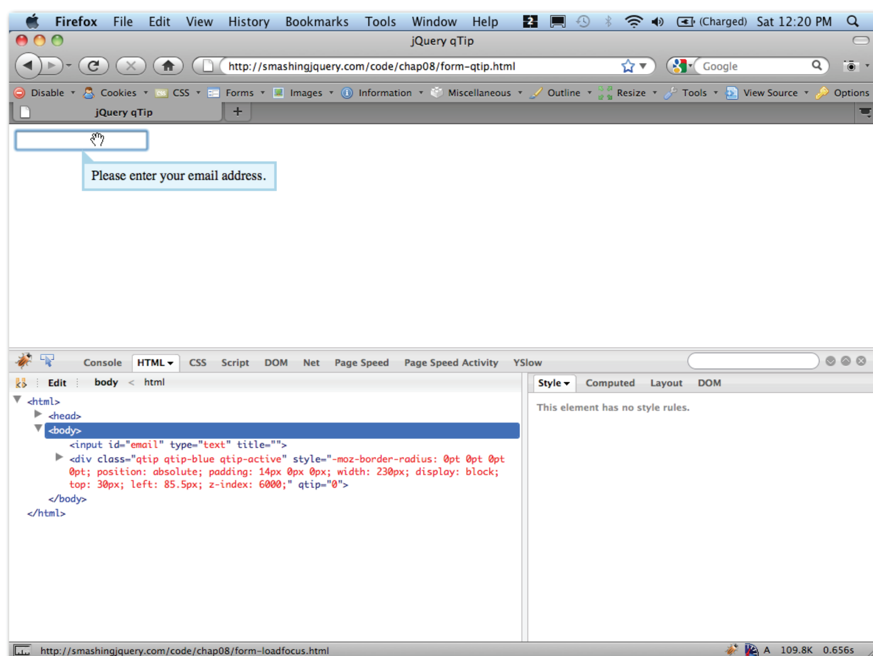


图8-17　使用qTip显示表单提示信息

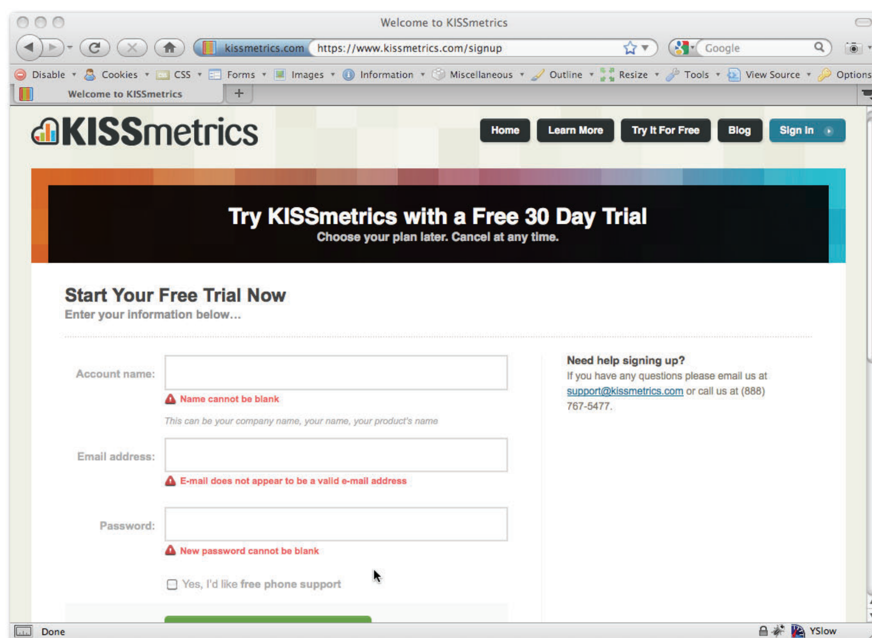
8.11.3　使用jQuery Validate插件验证表单

jQuery Validate插件自2006年7月问世以来，已成为至今可用的最古老且支持最好的验证插件。它支持许许多多的选项，用于设置Web站点或应用程序中的表单验证，这些我在本章中已经讨论过。如果你只需要简单的表单验证，比如验证少数几个文本框，我建议你使用jQuery编写自己的脚本来解决问题。

Validate插件是相当健壮的解决方案，它内建了标准的正则表达式来验证电话号码、电子邮件地址、日期和信用卡号码等。它通过许多事件（如submit、keypress和focus）把出错信息和提示信息直接存放在DOM中。Validate插件支持很广泛的API，支持对数据验证方法的完全定制。我们也可以选择“开箱即用”（内建的）的（验证）方法。

对那些需要姓、名、地址、城市、州、邮政编码及唯一用户名和（至少5位字符长且混用大小写字母的）密码的注册表单来说，Validate插件再合适不过了。一个长长的表单，不同的输入框需要不同的规则，而我们希望全程帮助用户完成这个表单。

图8-18演示的是一个使用了Validate插件的表单。如果用户单击了Submit按钮而没有正确填写所有的必填项，每个必填项之后都会出现提示文本，指出这些项是必填的。



由KissMetrics.com许可复制

图8-18　一个使用了Validate插件的表单

8.11.4　为联系人表单添加简单验证

使用Validate插件，我们可以节省下大量的时间，因为不必为每个输入框编写验证脚本。在下面的解决方案中，我演示了如何只使用一行jQuery代码和对表单的小小调整就使用上Validate插件提供的默认验证方法。

(1) 准备联系人表单，其中包含姓名、电子邮件地址、电话号码、信息栏和提交按钮。

```
<form id="contact-form">
    <ul>
        <li><label>Name</label>
        <input type="text" id="name" name="name"/>
        </li>
        <li><label>Email</label>
        <input type="text" id="email" name="email"/>
        </li>
        <li><label>Phone</label>
        <input type="text" id="phone" name="phone"/>
        </li>
        <li><label>Message</label>
        <textarea name="message" id="message"></textarea>
        </li>
        <li><input type="submit"/></li>
    </ul>
</form>
```

(2) 当使用jQuery插件时，我们总要先把插件包含到页头。它的包含位置应该总是刚好位于jQuery库之后，并且位于你自己编写的任何引用该插件的jQuery代码之前。

```
<script src="js/jquery.validate.min.js" type="text/javascript"></script>
```

(3) 在document的ready事件处理函数中，选中#contact-form元素并调用.validate()方法。

```
$(document).(function() {
    $("#contact-form").validate();
});
```

就这么简单！验证插件现在已经应用于你的Web站点。然而，如果在Firefox浏览器中访问这个页面并单击Submit按钮，你不会发现任何验证行为（见图8-19），这是因为我们还没有配置这个插件。

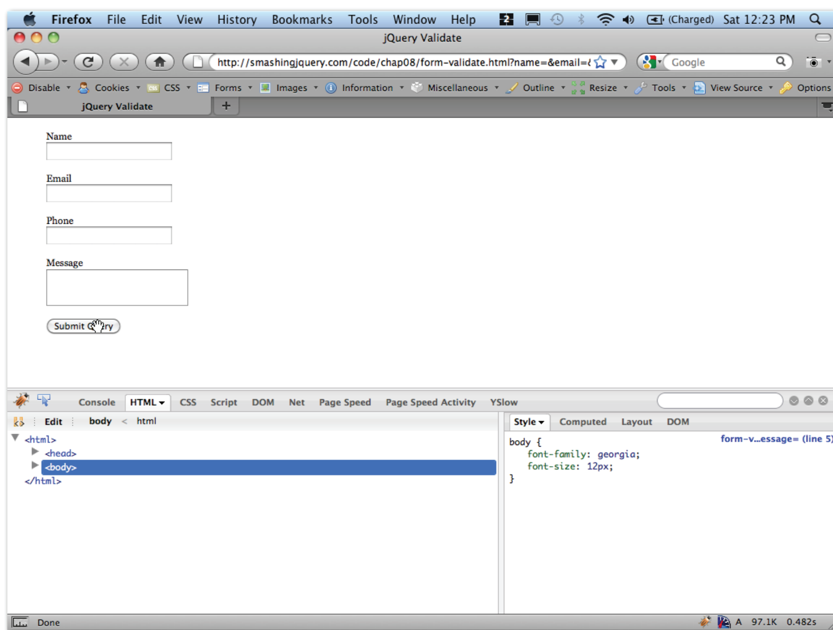


图8-19　提交表单未能触发任何验证行为

我们可以相当容易地使用class为表单添加验证行为。这是最快也最容易的方法，不过我们还不能控制提示信息或者其他像文本框最少需要5个字符、最多不超过10个字符之类的高级规则。那都是些非常棒的“开箱即用”功能。我会在下一个教程中详细介绍这些高级选项。

- 添加 class="required" 给那些必须填写的项。
- 添加 class="email" 给电子邮件地址输入框。
- 添加 class="digits" 给那些只允许输入数字的文本框。

(4) 调整表单代码，为表单域添加验证class：

```
<form id="contact-form">
    <ul>
        <li><label>Name</label>
        <input type="text" id="name" name="name" class="required"/>
        </li>
        <li><label>Email</label>
        <input type="text" id="email" name="email" class="required email"/>
        </li>
        <li><label>Phone</label>
        <input type="text" id="phone" name="phone" class="required digits"/>
        </li>
        <li><label>Message</label>
        <textarea name="message" id="message" class="required"></textarea>
        </li>
        <li><input type="submit"/></li>
    </ul>
</form>
```

给这些input元素添加完这些required／email／digit类之后，提交表单，就会看到如图8-20所示的提示出错的验证信息。如果你开始填写一个电子邮件地址，提示信息就由“This field is required”（该项必须填写）变更为“Please enter a valid email address”（请输入一个合法的电子邮件地址）。信息随着你的键入而变，一旦你键入了正确的电子邮件地址，表单也就不再显示提示信息。

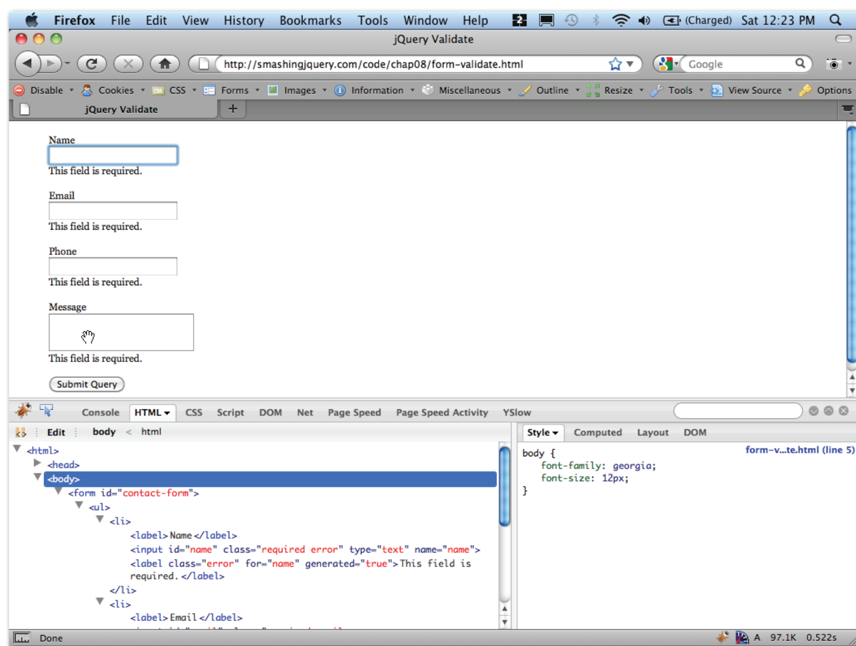


图8-20　使用Validate插件之后提交表单，错误信息随即显示在表单上

参考表8-3查阅Validate插件完整的选项列表。

表8-3　Validate插件选项

选　项　名	描　　述
required	必须填写一个值
minlength	要求的最小长度
maxlength	要求的最大长度
email	要求是一个合法的电子邮件地址
url	要求是一个合法的URL
date	要求是一个合法的日期
number	要求是一个合法的十进制数
digits	要求是合法的数字
creditcard	要求是合法的信用卡号码
accept	要求某种文件扩展名
equalTo	要求两个元素的值相等，经常用于密码和确认密码

注：要查阅完整的文档，请访问http://docs.jquery.com/Plugins/Validation。

下面列出了设置这些选项的两种方法。

- 第一种，使用类在表单中嵌入这些选项：

```
<input type="text" id="email" name="email" class="required email"/>
```

- 第二种，以JSON对象的形式传递这些选项给.validate()：

```
rules: {
  email : {
    required:true
  }
}
```

8.11.5　在联系人表单中使用高级验证规则并自定义提示信息

Validate插件支持可用于任意表单的许多极其灵活的选项，以确保所有的数据都被检查，并且在用户键入的信息不正确时及时提醒。本节，我们接着使用上一个例子中的表单代码，展示如何使用Validate API的高级选项来验证表单项并显示自定义提示信息（参见图8-21）。

(1) 为使用Validate插件生成的表单及错误信息设置样式。

```
body{font-family:georgia; font-size:12px}
label{display:block}
ul{list-style-type:none}
ul li{margin:15px 0}
.error, .notice, .success{padding:.8em; margin-bottom:1em; border:2px solid #ddd}
.error{background:#FBE3E4; color:#8a1f11; border-color:#FBC2C4}
.notice{background:#FFF6BF; color:#514721; border-color:#FFD324}
.success{background:#E6EFC2; color:#264409; border-color:#C6D880}
#contact-form{width:400px}
```

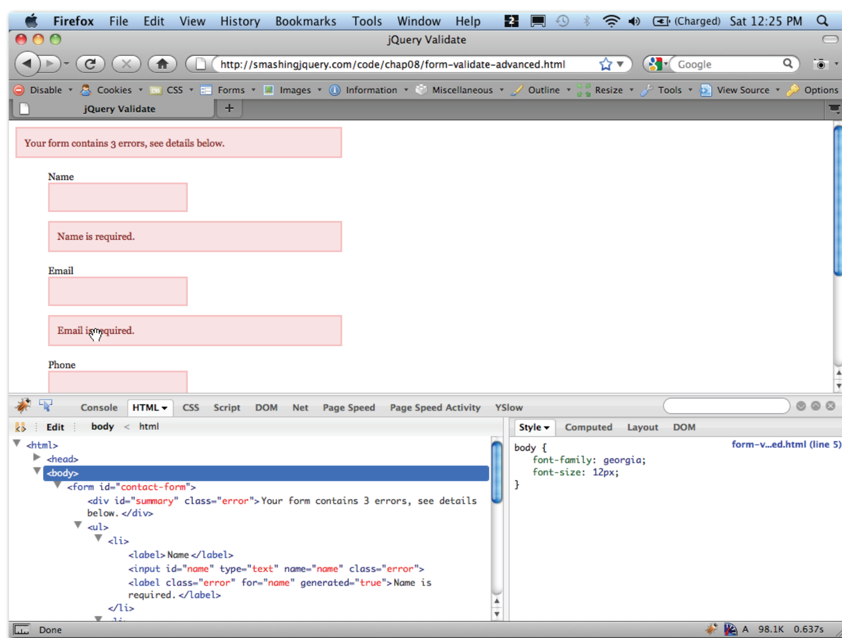


图8-21　具有高级规则和自定义信息的表单提交之后的显示情况

(2) 准备包含姓名、电子邮件地址、电话号码、信息栏和提交按钮的联系人表单：

```
<form id="contact-form">
    <ul>
        <li><label>Name</label>
        <input type="text" id="name" name="name"/>
        </li>
        <li><label>Email</label>
        <input type="text" id="email" name="email"/>
        </li>
        <li><label>Phone</label>
        <input type="text" id="phone" name="phone"/>
        </li>
        <li><label>Message</label>
        <textarea name="message" id="message"></textarea>
        </li>
        <li><input type="submit"/></li>
    </ul>
</form>
```

(3) 当使用jQuery插件时，我们总要先把插件包含到页头。它的包含位置应该总是刚好位于jQuery库之后，并且位于你自己编写的任何引用该插件的jQuery代码之前。

```
<script src="js/jquery.validate.min.js" type="text/javascript"></script>
```

(4) 选中`#contact-form`元素并调用`.validate()`方法。

```
$("#contact-form").validate();
```

(5) 首先为需要验证的表单元素设置高级规则。这些规则以JSON对象的形式传递给.validate()方法。我们使用这些input元素的name属性引用这些元素。姓名、电子邮件地址和电话号码是必须的。电子邮件地址必须被验证为真实地址。

```
$("#contact-form").validate({
  rules: {
    name: "required",
    email: {
      required: true,
      email: true
    },
    phone: "required"
   }
});
```

(6) 接下来，设置我们希望显示的自定义信息。我们想设置几个就设置几个。如果我们不设置自定义的错误信息，就会使用默认信息“This field is required”。使用和规则相同的格式，我们设置messages分支。在大括号中，添加每条信息时使用name属性加一个冒号，后面跟着希望使用的信息字符串。

```
$("#contact-form").validate({
  rules: {
    name: "required",
    email: {
      required: true,
      email: true
    },
    phone: "required"
  },
  messages: {
    name: "Name is required.",
    email: "Email is required.",
    phone: "Phone is required."
  }
});
```

瞧见没？在Firefox中载入这个页面，然后提交表单，所有表单项都被验证并且显示你刚刚创建的自定义提示信息。就这么简单。使用Validate插件的好处在于它运行在客户端，因此没必要刷新页面。在激活出错信息之后，如果你重新键入数据，插件脚本会实时验证你键入的数据。这为这款迷人的插件又加一分！

如果肯花些时间把玩这款插件，你还能得到更好的结果。如果你在使用这款插件时遇到任何问题，可阅读它的文档或者通过谷歌查找答案。很可能有人已经遇到过类似问题，更何况这款插件已经问世长达5年并且仍然有着良好的支持。

Part 4 第四部分

jQuery 高级技术

本部分内容

第 9 章

Ajax与动态数据处理

jQuery能够并且擅长维护DOM。你或许不知道jQuery还有许多支持Ajax操作的方法。许多人搞不清Ajax是怎么回事。随着Web 2.0的兴起Ajax这个术语越来越流行。Web设计师、开发者和市场营销人员都声称会用Ajax，然而Ajax到底什么意思？Ajax意味着我们能够设计、开发和运营支持幕后往返传递数据的Web站点和应用程序，让我们能够为顾客提供更加丰富的用户体验。

本章，我将介绍如何使用jQuery通过Ajax请求服务器端数据。我将使用前面章节编写的代码示例，演示如何添加动态内容，以生成自更新的组件。我不会讲述如何设置服务器端组件，因为需要像PHP或ASP.net那样的后端编程语言知识，而这超出了本书的范围。我会演示如何与几个流行的第三方服务API交互，以便使用jQuery通过这些API获取远端数据，并把这些数据呈现在站点上。

9.1 Ajax 揭秘

Ajax代表异步JavaScript与XML。还有一个较少人知道的术语XHR，表示XML HTTP请求。如果是一个Firebug（Firefox插件）老手，那你很可能已经通过Firebug的XHR标签看到过Ajax请求。如图9-1所示，XHR是Ajax请求的一种。

Ajax是指在不需要刷新（重新加载）页面的情况下，允许客户端应用程序传递数据给服务器并从服务器端获取数据的一组模式和技术。表面上看来，Ajax在幕后创建了一种平滑的数据流，这也是它名字中含有一个A（表示“异步”）的原因。它以异步的方式和服务器对话（交换数据）。Ajax请求以POST或GET请求的方式完成。

有些时候，Ajax请求也能够以同步方式执行。例如，有时我们需要载入一个外部配置文件，在确定配置文件成功载入之前，我们不希望执行后面的脚本。不过，本章我们只关心异步请求。

Facebook站点大量地使用了Ajax技术。在Facebook上登录后，你通常会看到朋友们的最新状态和最近的活动。左边是一个在线朋友列表，这是通过向服务器发起Ajax请求不断检查你的朋友中谁在线上实现的。或许你的一位朋友注意到你在线，于是启动了一个聊天会话——这又是一个Ajax请求。当你和朋友聊天时，另两位朋友更新了他们的状态，于是一条新消息出现在你的新鲜事顶部，提醒你有了新消息。所有这些事件都是通过Ajax请求完成的，你一次都不用刷新或者重新载入页面。这实在太棒了！

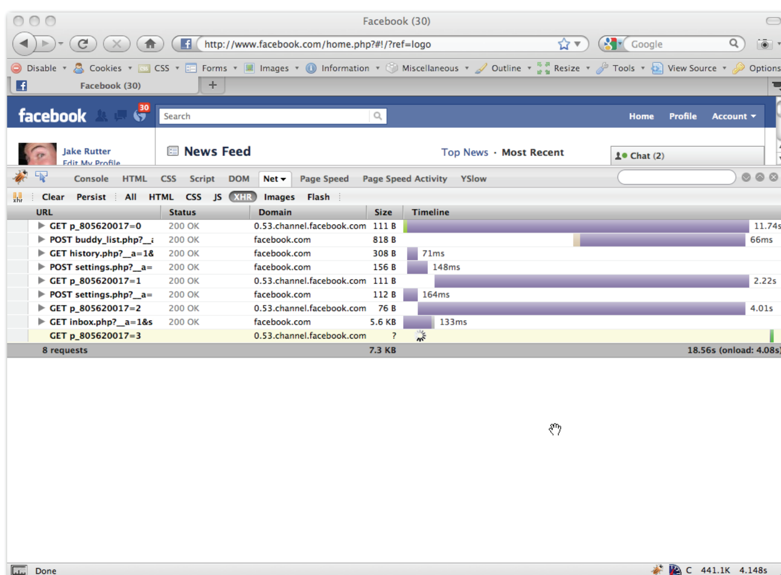
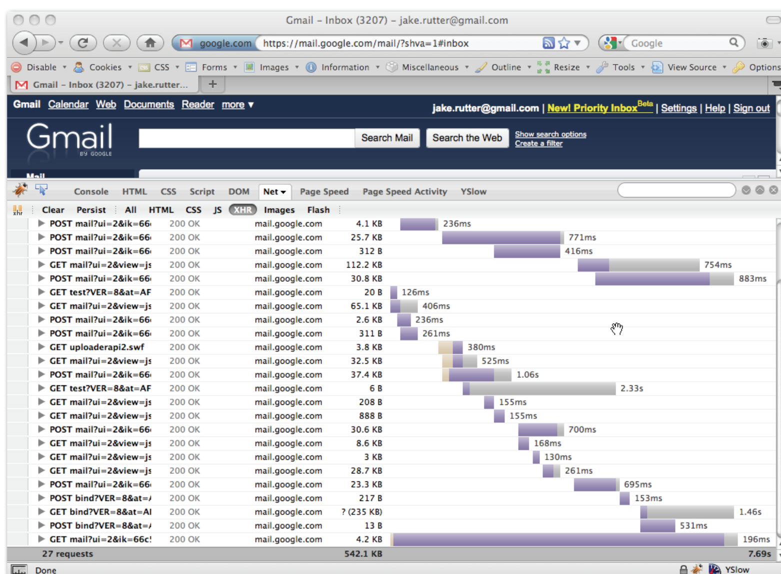


图9-1　Firebug插件展示的Facebook站点中的XHR请求

Gmail整合了很多Ajax功能，比如内建的实时聊天客户端、通过Gmail账号打电话的功能、移动信件的拖放功能，还有自动完成收信人账号等功能。Gmail是又一个把Ajax用到极致的产品（参见图9-2）。



© 2010，Google

图9-2　Firebug呈现的由Gmail发起的XHR请求

9.2　在页面上动态载入内容

我们可以使用jQuery的.load()方法，从服务器上载入存放在其他位置的HTML内容。.load()方法接受内容所在的URL作为参数，且需要以相对路径[1]（对当前页面来说）引用这个URL。.load()方法只能从存放在同一个服务器或域中的文件中载入数据。

如果希望载入其他域名下的内容，我们需要向Web服务、API或者支持JSON-P的Web服务器发起JSON-P请求（我会在本章后面讲解有关JSON-P的内容）。

```
$(selector).load(URL)
```

如果希望在内容载入完成后做些什么事情，我们可以提供一个回调函数给.load()方法，既可以是一个匿名函数，也可以是一个具名函数。

```
$(selector).load(URL, function(){
alert('The content was loaded');
})
```

如果选择器（selector）对应的元素不存在，则不会载入任何内容。

9.2.1　载入全部内容

如果希望载入一个独立HTML文件的全部内容，我们需要先选中存放内容的目标元素，然后以相对路径表示的HTML文件URL为参数调用.load()方法。

在下面的例子中，我利用一些由H2和P元素构成的示意内容准备了一个HTML页面。我还有一个HTML页面，其中有一个空的div#content元素，我计划为它载入内容。我使用.load()方法将内容“拖”入#content元素并显示在页面上（参见图9-3）。

(1) 创建一个用来载入的HTML文档。如果我们不为这些内容设置任何CSS样式，它就会继承父页面的样式。把这个文件保存为ajax-content.html。

```
<!doctype html>
  <head>
    <title>jQuery Content</title>
    <style>
    body {font-family:georgia;font-size:12px;}
    </style>
  </head>
  <body>
  <h1>Dolor Neque Placerat Sem Lacus Senectus</h1>

  <h2>Posuere Eleifend Amet</h2>
  <p>Scelerisque. Vivamus at interdum aliquam turpis euismod adipiscing dolor nec fusce
nulla amet facilisis fusce montes donec enim integer habitant euismod dignissim sodales
eu dui Lacus <em>potenti</em> gravida gravida. Amet cum. Accumsan hac.</p>

  <p>Ligula. Sodales, suscipit elementum. Faucibus tincidunt feugiat consectetuer cum
```

① 其实不必非要是相对路径，绝对路径也可以。

```
accumsan platea mauris augue <strong>curae;</strong> semper risus. Dapibus Scelerisque.
Ante proin leo. Dolor, arcu sociis mattis conubia nisi mi venenatis montes molestie
mi, per. Fermentum enim iaculis magnis nonummy.</p>

  <p>Rutrum <em>tortor</em> aptent vestibulum aliquet. Sollicitudin, mattis cras ac
accumsan bibendum pellentesque platea sociosqu parturient ad ligula. Sociis nisi mus
venenatis maecenas vel quisque. Volutpat turpis praesent tempus nulla.</p>
  </body>
</html>
```

(2) 在当前页面中添加一个#content元素。它用来存放.load()方法载入的内容。我们也可以把内容载入到某个具体的标签中，比如body标签。

```
<div id="content"></div>
```

(3) 在document的ready事件处理函数中添加一条语句，选中#content元素，然后调用.load()方法把ajax-content.html文件的内容载入页面。

```
$(document).ready(function() {
  $('#content').load('ajax-content.html');
});
```

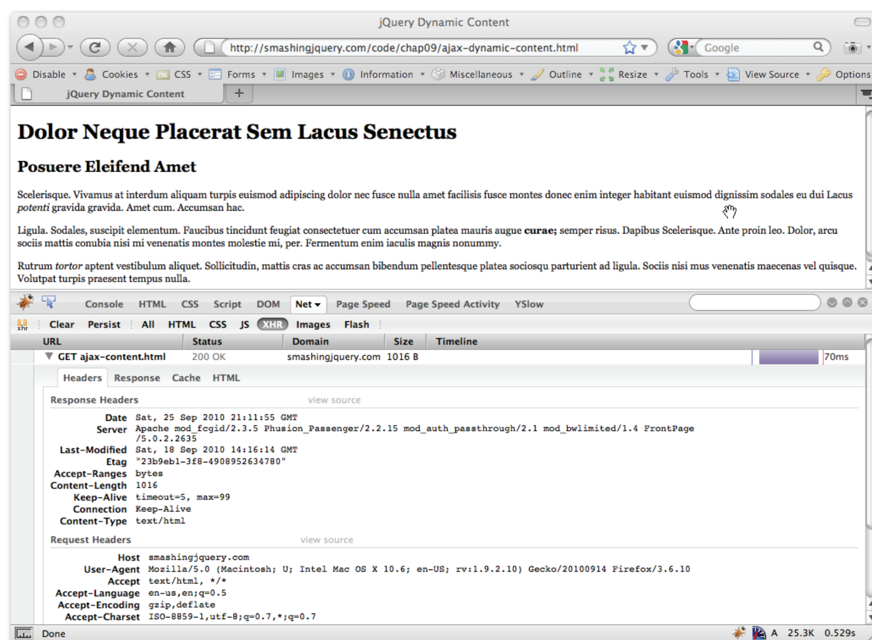


图9-3　使用.load()方法动态载入内容

9.2.2　在内容载入失败时处理错误

如果我们要载入的内容不存在怎么办？由于我们的用户看不到内容，他们或许会结束访问，

揣着一肚子怨言离开我们的站点。为了改进前面的脚本，我们需要传递一个检查文件是否存在的回调函数。如果文件不存在或者因为其他原因无法载入，我们可以显示一条消息。出错处理是一个好的编程习惯，我们应该在编写的每个脚本中做好错误处理。

`.load()`方法的回调函数支持3个参数（如表9-1所示）。利用`XMLHttpRequest`对象可得到Ajax请求的响应码，对不同的响应码我们执行不同的行为。

```
$(selector).load(URL, function(responseText, textStatus, XMLHttpRequest){
// 检查不同的响应码
})
```

表9-1　XHR 请求属性

属 性 名	描　　述
`responseText`	以字符串形式返回响应数据
`textStatus`	以XML数据形式返回响应数据
`XMLHttpRequest`	返回状态码，比如“404”（表示“Not Found”）或者“200”（表示“OK”）

来自www.w3schools.com/dom/dom_http.asp

接着前面的代码，我来讲解一下如何在回调函数中添加一点代码，使用表9-2列出的服务器响应码捕获任意的错误。图9-3展示的就是这个例子。服务器响应码来自Web服务器，通过HTTP协议传递给浏览器。这些响应码通过`XMLHttpRequest.status`进行传递。我们也可以通过Firebug的XHR面板查看这些响应码（参见图9-2，在Firebug中查看Gmail的Ajax请求）。

表9-2　常见服务器错误响应码

响 应 码	含　　义
200	成功
301	永久跳转
302	临时跳转
400	错误请求
401	未授权
403	禁止访问
404	未找到
500	服务器错误

(1) 为`.load()`方法添加另一个参数，即回调函数，传入这个函数支持的3个属性参数。

```
$('#content').load('ajax-content-1.html',
 function(responseText, textStatus, XMLHttpRequest){
});
```

(2) 在回调函数中添加一行`if/else`语句，检查`404`或`500`错误。如果错误发生，如图9-4所示，显示一条错误信息。否则，内容就正常载入，无需显示任何信息。

```
$('#content').load('ajax-content-1.html', function(responseText, textStatus,
  XMLHttpRequest){
  if (XMLHttpRequest.status == 404 || XMLHttpRequest.status == 500) {
    $('#content').html('There has been an error, please try again later.');
  } else {
  // 什么也不做
  }
      });
```

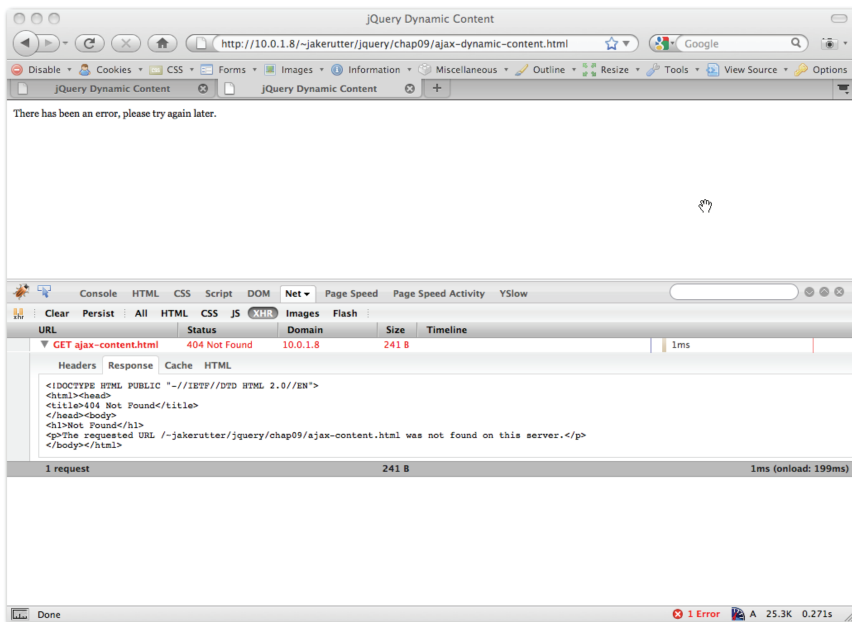


图9-4　当被请求内容不可用时，一条404错误信息显示出来

如果载入的内容很多，最好显示一个载入动画，以便让用户知道数据正在载入中。我们可以在.load()方法前添加一条语句，在#content元素中添加一个表示载入中的动态gif图片（数据载入成功之后，就会替换这张图片）。如果载入的数据量很少，用户可能永远也没有机会看到这张图片，不过这是一个好的预留手段，有备无患，不用担心大文件载入或者缓慢的网络连接。

```
$('#content').html('<img src="images/loading.gif" alt=""/>');
$('#content').load('ajax-content-1.html', function(responseText, textStatus,
XMLHttpRequest)
  {
  if (XMLHttpRequest.status == 404 || XMLHttpRequest.status == 500) {
    $('#content').html('There has been an error, please try again later.');
  } else {
  // 什么也不做
  }
});
```

9

9.2.3　载入部分内容

如果我们只想载入外部HTML文件中的某一部分内容。在`.load()`方法中以参数形式传入ID、类或者标签名字就能做到这一点。

```
$(selector).load(URL class or id or tag name)
```

在下面这个例子里，我继续使用上文用过的由`H2`和`P`元素组成的示意内容页面。其中一个段落（p标签）具有`special`类。我还准备了一个内含一个空`div#content`元素的页面，用它来存放jQuery载入的内容。在本例中，我仅仅载入示意页面中那个拥有`special`类的段落（参见图9-5）。

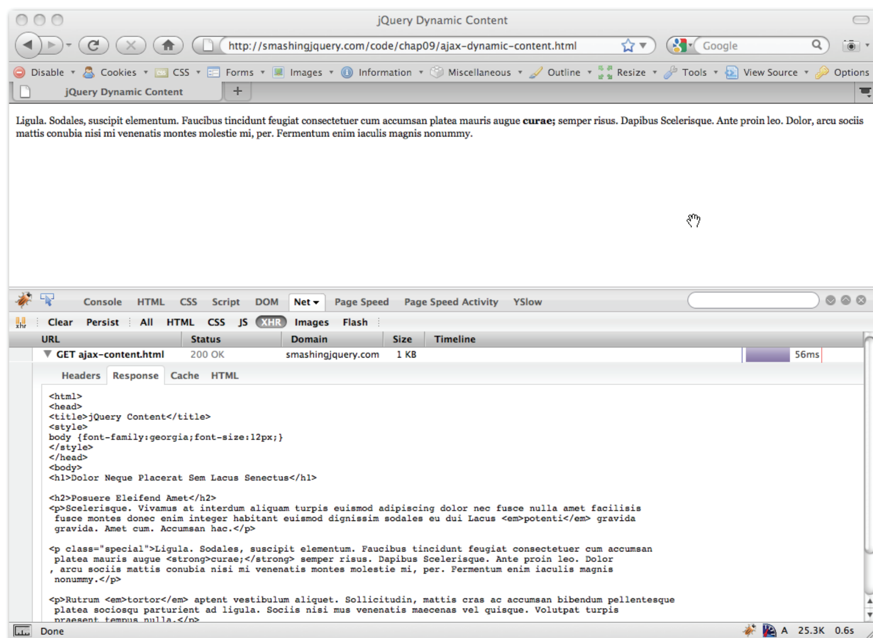


图9-5　只有`special`类对应的内容被载入当前页面

(1) 创建被载入的HTML页面内容，给第二个段落加上`special`类。在第二步中要通过这个类抓取这段内容。将这个文件另存为ajax-content.html。

```
<!doctype html>
  <head>
    <title>jQuery Content</title>
    <style>
    body {font-family:georgia;font-size:12px;}
    </style>
  </head>
  <body>
  <h1>Dolor Neque Placerat Sem Lacus Senectus</h1>
```

```
  <h2>Posuere Eleifend Amet</h2>
  <p>Scelerisque. Vivamus at interdum aliquam turpis euismod adipiscing dolor nec fusce
nulla amet facilisis fusce montes donec enim integer habitant euismod dignissim sodales
eu dui Lacus <em>potenti</em> gravida gravida. Amet cum. Accumsan hac.</p>

  <p class="special">Ligula. Sodales, suscipit elementum. Faucibus tincidunt feugiat
consectetuer cum accumsan platea mauris augue <strong>curae;</strong> semper risus.
Dapibus Scelerisque. Ante proin leo. Dolor, arcu sociis mattis conubia nisi mi venenatis
montes molestie mi, per. Fermentum enim iaculis magnis nonummy.</p>

  <p>Rutrum <em>tortor</em> aptent vestibulum aliquet. Sollicitudin, mattis cras ac
accumsan bibendum pellentesque platea sociosqu parturient ad ligula. Sociis nisi mus
venenatis maecenas vel quisque. Volutpat turpis praesent tempus nulla.</p>
  </body>
</html>
```

(2) 在当前页面中添加一个#content元素。如果被载入的页面中没有选择器对应的元素，也就没有任何内容会插入到#content元素内（内容加载仍会发生）。

```
<div id="content"></div>
```

(3) 在document的ready事件处理函数中添加一条语句，选中#content元素，然后载入ajax-content.html到页面。注意.load()方法参数中的special类，是它决定了只有special类对应的内容才会被添加到页面中。

```
$(document).ready(function() {
    $('#content').load('ajax-content.html .special');
});
```

9.3 使用 GET 和 POST 方法提交表单

GET和POST请求非常相似，它们都能把表单中的数据传递给服务器端进程。两种方法各有各的优点，各有各的缺点。通常都是后端程序员定义好请求类型，前端开发人员和设计师只管使用就行了。

1. 理解GET请求

GET请求以查询字符串的形式把数据传递给服务器端进程。表单数据通过查询字符串从一个Web页面或应用程序传递到另一个Web页面或应用程序。服务器端进程总是在URL中拾取键值对，并依据URL中的查询字符串直接改变页面内容 。

这里是一个查询字符串的例子：

```
http:// www.website.com?keyword=product&page=5&size=5&color=brown
```

通过查询字符串传递参数，风险在于查询字符串太容易被用户修改。如果服务器端脚本没有适当的安全预防机制，黑客就能入侵系统并进而大肆破坏你的站点或应用程序。出于安全方面的考虑，尽量不要使用GET请求传递类似财务信息、密码或其他要传递的数据及显示给用户的数据。就像图9-6所示的谷歌搜索结果页那样，GET请求可以保存为书签或者分享给其他人。

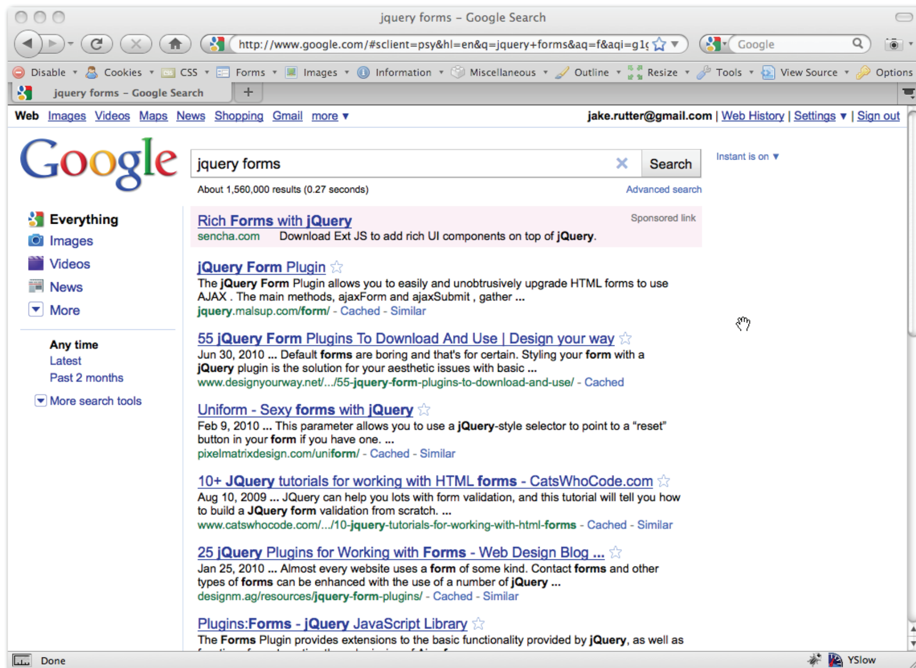


图9-6　使用了GET请求的谷歌搜索结果页

2. 理解POST请求

POST请求与GET请求不同，它在“幕后”发送数据给服务器端进程，这使得POST方法比GET更安全，特别是在传递敏感数据时。与GET请求相比，POST请求能够一次传递大量的数据给服务器端程序。（而GET请求受URL长度限制，一次只能传递较少的数据。）

如果提交了一个POST请求并尝试刷新页面，浏览器会提示你页面需要重新发送数据到服务器。这是由POST请求结构化的特点决定的。

在jQuery中使用`.ajax()`方法处理POST和GET请求。这个方法支持许多参数，用于控制数据如何发送到服务器以及如何接收数据。

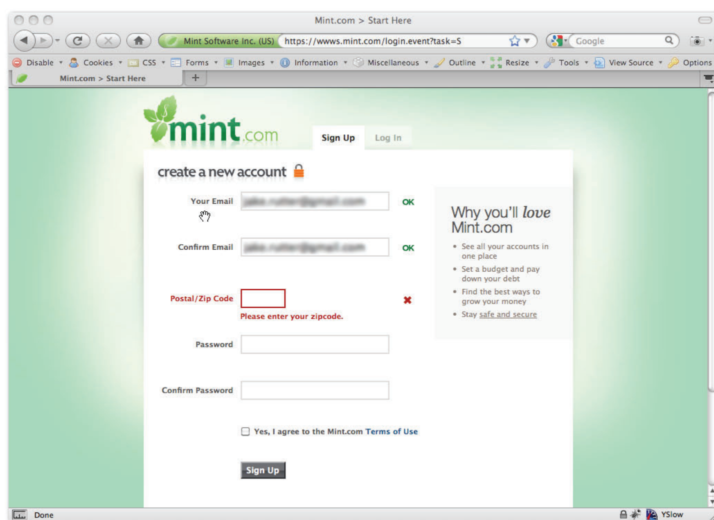
```
$.ajax ({
   type: 'POST',
   url: url,
   data: data,
   success: success,
   dataType: datatype
});
```

如果我们用原生JavaScript实现这个方法，最终将会得到一段与下面类似的代码。我们需要为不同的浏览器编写不同的`XMLHttpRequest`方法。jQuery以一个又棒又干净的`.ajax()`方法替我们处理好了这一切。

```
function loadXMLDoc() {
  if (window.XMLHttpRequest){
      // 针对 IE7+、Firefox、Chrome、Opera、Safari的代码
      xmlhttp=new XMLHttpRequest();
  } else {
      // 针对IE6、IE5的代码
      xmlhttp=new ActiveXObject("Microsoft.XMLHTTP"); }
      xmlhttp.onreadystatechange=function(){
      if (xmlhttp.readyState==4 && xmlhttp.status==200) {
          document.getElementById("myDiv").innerHTML=xmlhttp.responseText; }
    }
    xmlhttp.open("GET","ajax_info.txt",true);
    xmlhttp.send();
}
```

3. 使用POST方法实现无页面刷新的联系人表单提交

由于联系人表单通常包含个人信息及需要保密的敏感信息，因此特别适合使用POST请求提交。POST请求也常常用于登录和注册表单，比如图9-7所示的Minit.com的注册表单页就使用了POST请求。



由Mint.com许可复制

图9-7　Mint.com使用POST请求处理账号注册表单

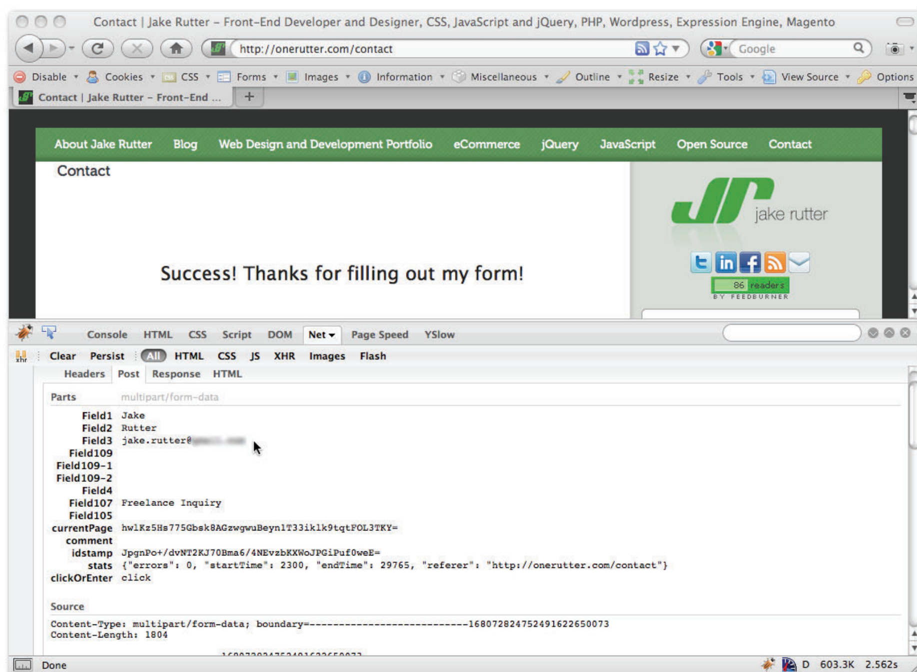
在jQuery中使用POST请求提交数据极其容易。和绝大多数jQuery方法一样，使用POST请求提交数据也有两种方式可用，一是标准方式，一是快捷方式。下面两段代码中，第一段使用`.post()`方法，这是标准方式。第二种使用`.ajax()`方法，由于`.ajax()`方法能够处理多种请求，所以它是快捷方式，也是我们推荐使用的方式。[①]

① 作者和我们的思维方式有所不同，作为一名中国人，我一直觉得`$.get`/`$.post`这些方法才是快捷方法（更少的参数使用，更方便），而`$.ajax()`因为要设置大量参数反而较少用。

```
$.post(url, [data], [success], dataType);

$.ajax({
  url: url,
  data: data,
  success: success,
  dataType: dataType
});
```

由于我们不能通过查询字符串看到参数，在浏览器中调试POST提交很有挑战性。我们可使用Firebug监视XHR请求并查看它们“幕后”传递的参数，见图9-8。



由Mint.com许可复制

图9-8　“联系我们”表单使用POST请求提交参数后的画面

(1) 准备用来提交给服务器的HTML表单。由于`input`元素的ID将在下一步操作中使用，因此为每个`input`元素选择不同的ID至关重要。

```
<form id="contact-form">
    <ul>
        <li><label>Name</label>

        <input type="text" id="name" name="name"/>
        </li>
        <li><label>Email</label>
        <input type="text" id="email" name="email"/>
```

```
        </li>
        <li><label>Phone</label>
        <input type="text" id="phone" name="phone"/>
        </li>
        <li><label>Message</label>
        <textarea name="message" id="message"></textarea>
        </li>
        <li><input type="submit" id="submit"/></li>
    </ul>
</form>
```

设想下面的场景：用户登录我们的站点，填写表单，然后使用Ajax请求把数据悄悄地提交到服务器。

(2) 选中#submit按钮，为它绑定click事件处理函数。添加一行return false语句，阻止提交按钮的默认行为。

```
$('#submit').bind('click', function(){
  return false;
});
```

(3) 接下来，引入4个变量——nameVal、emailVal、phoneVal和msgVal，并把这些变量的值设置为相应input元素的值。这样我们就从表单中获取到即将传递给服务器的各个值。

```
$('#submit').bind('click', function(){
  var nameVal = $('#name').val();
  var emailVal = $('#email').val();
  var phoneVal = $('#phone').val();
  var msgVal = $('#message').val();
  return false;
});
```

(4) 在这些变量之后添加$.post()方法调用。为使$.post()方法正常工作，我们必须传入所有的表单参数。我们将上一步得到的参数值以键值对的形式传递给$.post()方法。

```
$('#submit').bind('click', function(){
  var nameVal = $('#name').val();
  var emailVal = $('#email').val();
  var phoneVal = $('#phone').val();
  var msgVal = $('#message').val();
  $.post("form.php",
  {name:nameVal,
  phone:phoneVal,
  email:emailVal,
  message:msgVal}
  );
  return false;
});
```

(5) 最后，不管是用它检查错误，还是在提交失败时显示错误信息给用户，我们总要引入一个回调函数。在$.post()方法中，表单参数之后，我们传入一个回调函数参数。图9-9展示的是

浏览器中下面的代码提交给服务器的参数。

```
$('#submit').bind('click', function(){
  var nameVal = $('#name').val();
  var emailVal = $('#email').val();
  var phoneVal = $('#phone').val();
  var msgVal = $('#message').val();
  $.post("form.php",
  {name:nameVal,
  phone:phoneVal,
  email:emailVal,
  message:msgVal}, function(data) {
    alert('Successful Submission');
  }
);
});
```

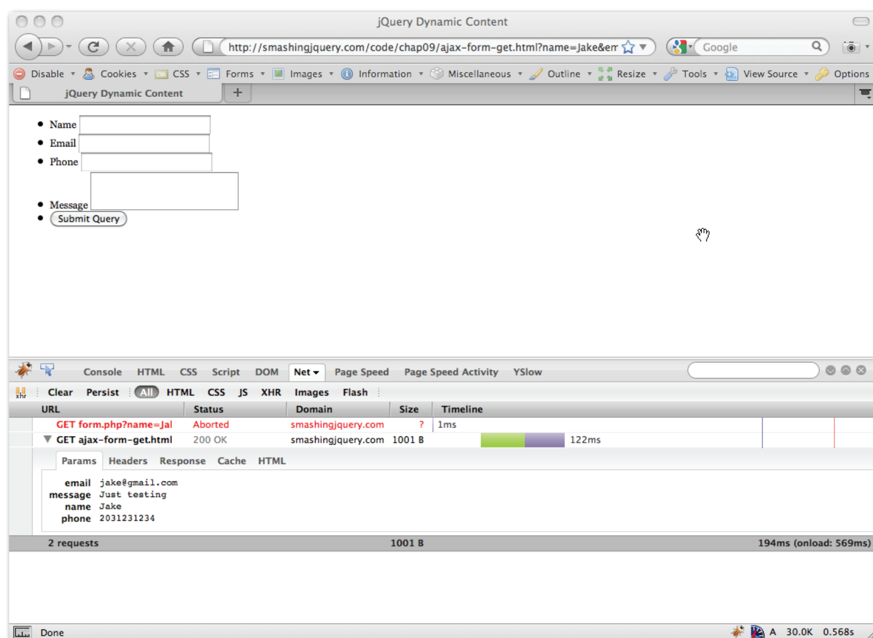
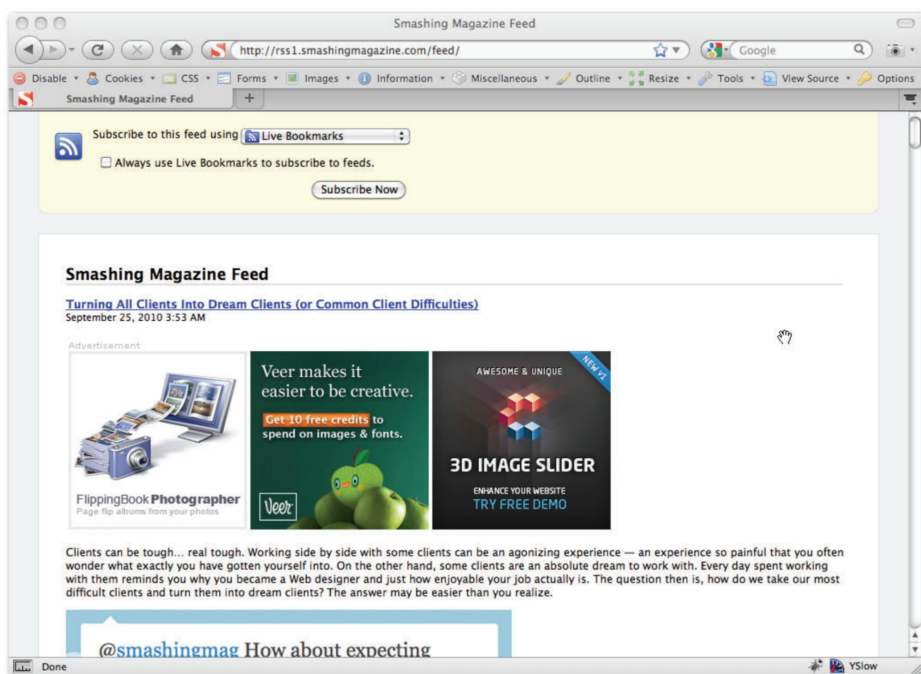


图9-9　表单提交之后，参数已被提交到服务器

9.4　操作 XML 数据

XML表示可扩展标记语言，问世于1996年，算不上新技术。XML是跨平台的公共标准，我们可以使用它定义个性化的数据结构。XML广泛用于各种应用程序，如提交银行数据、创建RSS（简单资讯聚合）种子等。可以说XML无处不在，是一种值得信赖的数据传递方法。图9-10展示的是来自Smashing Magazine（www.smashingmagazine.com）的RSS订阅源。



Smashing Media GmbH，由Sven Lennartz 和Vitaly Friedman创建

图9-10　Smashing Magazine提供的RSS订阅源

想象这样的场景：你的主页上有一些旋转木马的图片。这些图片来自一个商业用户，他负责上传并维护这些图片，为每张图添加标题、描述及一个URL（用户单击图片将被导航至该URL）。你可以找一个程序员一起创建图片数据的XML数据源，然后每日进行更新。然后你就可以使用jQuery对订阅源发起GET请求来生成主页的旋转木马图片。使用XML订阅源的好处是它支持像Flash、移动应用程序等前端平台。从某种意义上说，这样可用一种数据结构满足多种应用程序的需求，并且只需要程序员介入一次，完成XML数据源的开发。

使用XML的缺点是只能使用和你工作的站点同域的XML数据，并且代码臃肿。如果你希望和其他域名下的数据打交道，就需要在客户端使用JSON–P技术，我会在下一个解决方案中讲解这一技术。

我使用下面的XML代码创建了一个XML文件，并将在下面的XML解决方案中使用这个文件。我把这个文件保存为favoritebooks.xml，并把它和其他文件放到同一个目录下。后端程序员可以让服务器上的程序实时生成XML数据，相应地，我们可以获取这些数据并利用它们做事。

XML使用分层结构，可以描述成一棵拥有父子分支的家谱树，其中的元素又称为节点。下面的代码是一个XML示例，图9-11展示的是该XML在浏览器中的呈现。

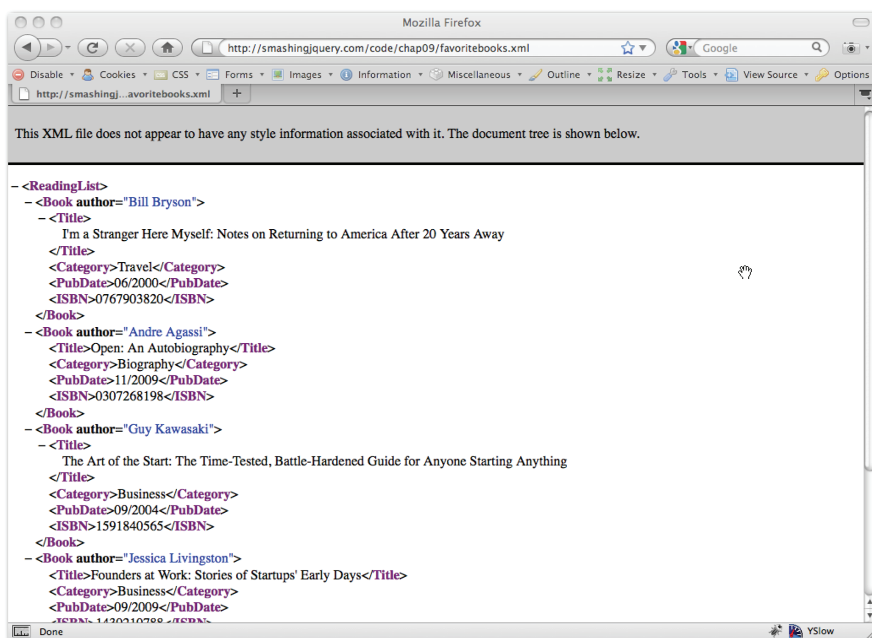


图9-11　在Firefox浏览器中访问前面的XML代码

```
<?xml version="1.0" encoding="utf-8" ?>
<ReadingList>
  <Book author="Bill Bryson">
    <Title>I'm a Stranger Here Myself: Notes on Returning to America After 20 Years
    Away</Title>
    <Category>Travel</Category>
    <PubDate>06/2000</PubDate>
    <ISBN>0767903820</ISBN>
  </Book>
  <Book author="Andre Agassi">
    <Title>Open: An Autobiography</Title>
    <Category>Biography</Category>
    <PubDate>11/2009</PubDate>
    <ISBN>0307268198</ISBN>
  </Book>
  <Book author="Guy Kawasaki">
    <Title>The Art of the Start: The Time-Tested, Battle-Hardened Guide for Anyone
    Starting Anything</Title>
    <Category>Business</Category>
    <PubDate>09/2004</PubDate>
    <ISBN>1591840565</ISBN>
  </Book>
  <Book author="Jessica Livingston">
    <Title>Founders at Work: Stories of Startups' Early Days</Title>
    <Category>Business</Category>
```

```
    <PubDate>09/2009</PubDate>
    <ISBN>1430210788</ISBN>
  </Book>
  <Book author="Jason Fried">
    <Title>Getting Real: The smarter, faster, easier way to build a successful web
    application</Title>
    <Category>Business</Category>
    <PubDate>09/2009</PubDate>
    <ISBN>0578012810</ISBN>
  </Book>
</ReadingList>
```

9.5　解析内部 XML 数据并生成 HTML

使用前面例子中的XML，我们使用jQuery通过Ajax请求载入XML数据，然后解析XML生成HTML内容。我们将使用`$.ajax()`方法，不过你也可以使用快捷方法`$.get()`。载入XML非常容易，只要寥寥几行jQuery代码就能完成。这是两个步骤：首先，加载XML；其次，在回调函数中解析得到的XML数据（回调函数仅在XML成功加载后执行）。

这个脚本最终的运行结果是一个包含我最喜欢的5本书的无序列表，如图9-12所示。

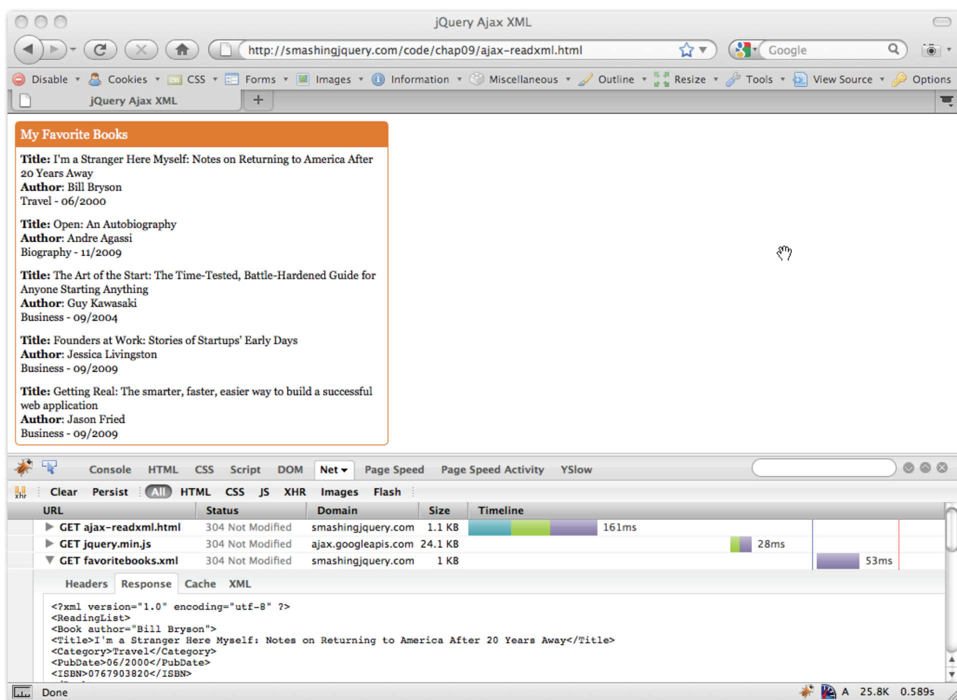


图9-12　得到XML并生成My Favorite Books（我最喜欢的书）组件

(1) 首先，我们需要准备存放书籍无序列表的HTML。创建一个无序列表ul#books。

```
<ul id="books">
<h1>My Favorite Books</h1>
</ul>
```

(2) 接着调用$.ajax()方法并提供4个参数：type参数代表请求类型，我们在这里使用GET方式；dataType参数是XML；url参数是favoritebooks.xml；回调函数是parseXML。

```
$.ajax({
type: "GET",
dataType: "XML",
url: "favoritebooks.xml",
success: parseXML
});
```

(3) 生成一个空白的回调函数parseXML。它有一个参数xml，代表收到的XML数据。不管回调函数中是否定义了这个参数，jQuery都会自动传递参数给回调函数。参数的内容取决于回调函数的定义类型（success回调还是error回调）。

```
function parseXML(xml) {
}
```

(4) 在parseXML函数中把xml当成选择器，然后调用.find()方法和.each()方法找出每一个Book节点。

```
function parseXML(xml) {
  $(xml).find("Book").each(function(){
  });
}
```

(5) 创建4个变量——author、title、category和pubdate。使用不同的选择器得到需要的数据赋给这些变量。由于author是Book节点的属性，所以我们需要使用获取属性的方法.attr()得到。剩下的几项都可以通过用.find()方法搜索当前节点并抓取相应的文本得到。

```
function parseXML(xml) {
  $(xml).find("Book").each(function(){
    var author = $(this).attr('author');
    var title = $(this).find('title').text();
    var category = $(this).find('category').text();
    var pubdate = $(this).find('pubdate').text();
  });
}
```

(6) 设置所有这些变量之后，现在可以生成HTML了。使用$('
')生成一个li元素，然后使用.html()方法，把由标签字符串和变量拼接而成的HTML内容纳入这个li元素，最后将它追加到ul#books中去。

```
function parseXML(xml) {
  $(xml).find("Book").each(function(){
    var author = $(this).attr('author');
    var title = $(this).find('title').text();
    var category = $(this).find('category').text();
```

```
    var pubdate = $(this).find('pubdate').text();
    $('<li></li>').html('<b>Title:</b> '+title +'</br><b>Author</b>: '+ author
      +'</br>'+category +' - '+pubdate).appendTo('#books');
  });
            }
```

9.6　操作 JSON 数据

JSON表示JavaScript对象表示法，它类似于XML，但只服务于JavaScript。[①]JSON允许我们以键值对的形式创建自己的数据结构。JSON和XML的相似之处在于它们都是层级结构，都支持Ajax处理。

与XML相比，JSON有一些先进之处，而且我个人认为JSON可读性更好。使用JSON时，我们无需使用标签定义结构，因为数据本身就定义了结构。使用这种更干净的方案，代码要比XML方案简洁得多，更容易开发，也更容易理解。网上有关JSON更好还是XML更好的争论此起彼伏，因此我刚刚发表的意见仅代表个人看法。

下面是一个JSON代码示例（参见图9-13）。我使用与前面XML例子完全相同的数据，便于大家对这两种类型的数据进行比较。第一印象就是JSON的可读性更好。这算不算一大优势值得商榷——毕竟绝大多数时候是你的脚本（而不是我们）读取这些JSON代码。

```
{"books":[
    {
    "title": "I'm a Stranger Here Myself: Notes on Returning to America After
      20 Years Away",
    "author": "Bill Bryson",
    "category": "Travel",
    "pubdate": "06/2000",
    "isbn": "0767903820"
    },
    {
    "title": "Open: An Autobiography",
    "author": "Andre Agassi",
    "category": "Biography",
    "pubdate": "11/2009",
    "isbn": "0307268198"
    },
    {
    "title": "The Art of the Start: The Time-Tested, Battle-Hardened Guide for
     Anyone Starting Anything",
    "author": "Guy Kawasaki",
    "category": "Business",
    "pubdate":"09/2004",
    "isbn": "1591840565"
    },
    {
    "title": "Founders at Work: Stories of Startups' Early Days",
```

① 这种说法不确切，如今几乎所有的语言都支持JSON数据。

```
            "author": "Jessica Livingston",
            "category": "Business",
            "pubdate": "09/2009",
            "isbn": "1430210788"
            },
            {
            "title": "Getting Real: The smarter, faster, easier way to build a
            successful web application",
            "author": "Jason Fried",
            "category": "Business",
            "pubdate": "09/2009",
            "isbn": "0578012810"
            }
        ]
}
```

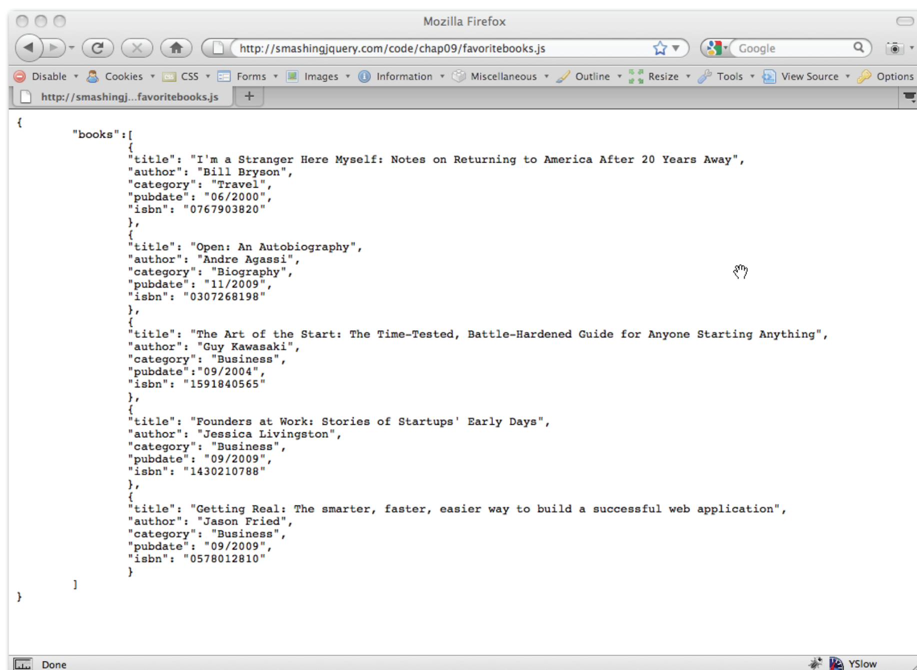


图9-13　前面的JSON代码显示在浏览器中

当我们撰写JSON代码时，可以使用验证工具来确保写出来的JSON代码是正确无误的。如果代码中有错误，当使用GET方法获取JSON数据到页面时，我们只会得到出错信息而非我们想要的数据。Douglas Crockford编写了一个名叫JSONLint（www.jsonlint.com）的JSON验证程序，如图9-14所示。它接受原始JSON代码或者JSON代码所在的URL。如果我们提供给这一验证工具的JSON数据有问题，就会得到有用的错误信息。图9-14展示的是我把前面的JSON代码交给JSONLint检验并验证通过的结果。

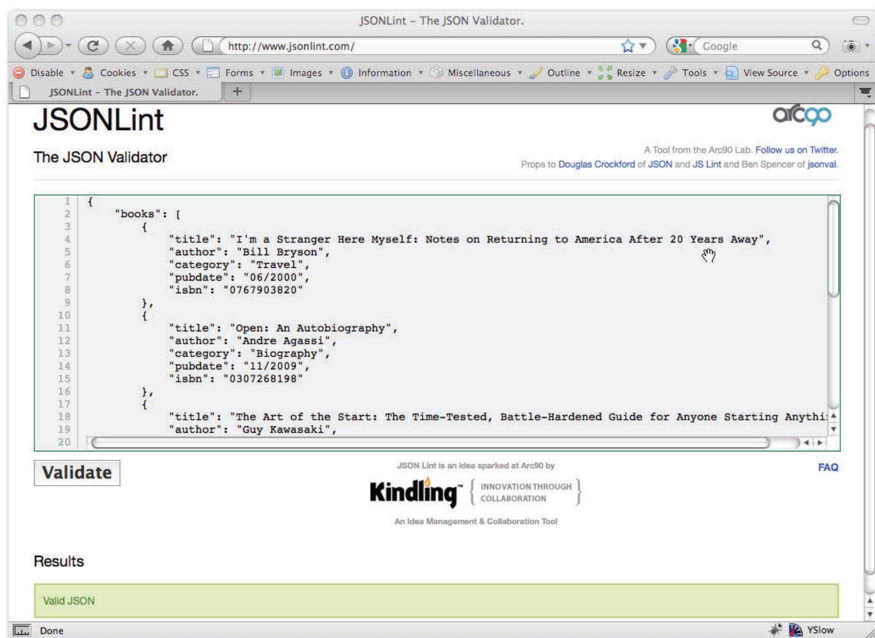


图9-14　Douglas Crockford的JSONLint验证工具

9.7　获取 JSON 数据并生成 HTML

使用前面例子中的JSON代码，我们使用jQuery通过Ajax请求载入JSON数据，接着像在页面上解析HTML那样解析JSON数据（参见图9-15）。[①]我们将使用`.ajax()`方法，不过你也可以使用快捷方法`$.getJSON()`。

使用jQuery获取JSON数据和获取XML数据非常相似，由于JavaScript原生支持JSON数据，因此只需要更少的步骤，这使得JSON成为一个性能更好的XML替代品。

这个脚本最终的运行结果是一个包含我最喜欢的5本书的无序列表，如图9-12所示。

(1) 首先，我们需要准备存放书籍无序列表的HTML。创建一个无序列表`ul#books`。

```
<h1>My Favorite Books</h1>
<ul id="books">
</ul>
```

(2) 接着调用`$.ajax()`方法并提供4个参数：`type`参数代表请求类型，我们在这里使用GET方式；`dataType`参数是`JSON`；`url`参数是`favoritebooks.json`；回调函数是`parseJSON`，我们将会在下一个步骤创建这个函数。

```
$.ajax({
```

① XML的解析类似HTML，JSON解析与HTML完全不同，作者此处的描述有误。

```
type: "GET",
dataType: "JSON",
url: "favoritebooks.json",
success: processJSON
});
```

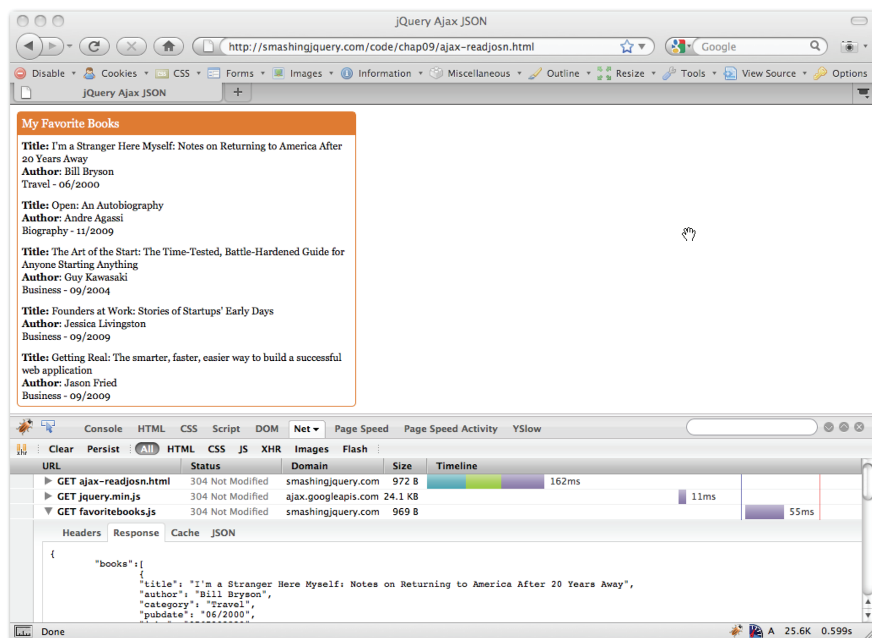


图9-15　得到JSON数据生成HTML并显示在页面上

(3) 编写一个空白的回调函数parseJSON。它有一个参数data，代表收到的JSON数据。

```
function processJSON(data) {
}
```

(4) 在parse JSON函数内使用$.each()方法创建一个循环，遍历data.books对象字面量中的数据。i是数组的索引，item则是相应的值。

```
function processJSON(data){
  $.each(data.books, function(i,item){
  });
}
```

(5) 在$.each()方法内[①]，使用$('
')动态生成一个li元素，然后使用.html()方法，把由标签字符串和变量拼接而成的HTML内容纳入这个li元素，之后把它追加到ul#books中去。我们使用item和键（名字）访问对应的值（数据），比如标题对应item.title。与XML相比，这是一大优点：我们不必创建中间变量就能直接访问数据项。

① 严格来说此处是指$.each()的回调函数内。

```
function processJSON(data){
  $.each(data.books, function(i,item){
  $('<li></li>').html('<b>Title:</b> '+item.title +'</br><b>Author</b>:
  '+ item.author +'</br>'+item.category +' - '+item.pubdate).appendTo('#books');
  });
}
```

两个解决方案的最终结果相同，而JSON方案使用更少的步骤，处理数据更快。生成JSON代码需要一点点技巧，因为它有着严格的语法：一个多余的逗号或大括号，就会让JSON数据彻底失效。

9.8　使用 Delicious API 接收 JSONP 数据以创建 Delicious 用户组件

Delicious（www.delicious.com，参见图9-16），是一家专注分享与发现Web书签的社交站点。在这家网站上我们可以收藏自己喜欢的站点并查看别人的收藏。Delicious站点发布于2003年9月，并于2005年卖给雅虎。这个站点的用户超过530万，一共收藏了超过1.8亿个书签。我曾经用过Delicious好几年，通过它了解人们喜欢什么链接，什么链接有用。它支持通过任意一台联网的计算机访问自己的书签，它还是一个发布自己项目的好地方，能传播自己的项目赢得更多的流量。Delicious针对许多浏览器开发了书签插件，无需访问Delicious站点就能随时保存书签，这为人们带来了极大的方便。这也是该应用如此流行的一个原因。

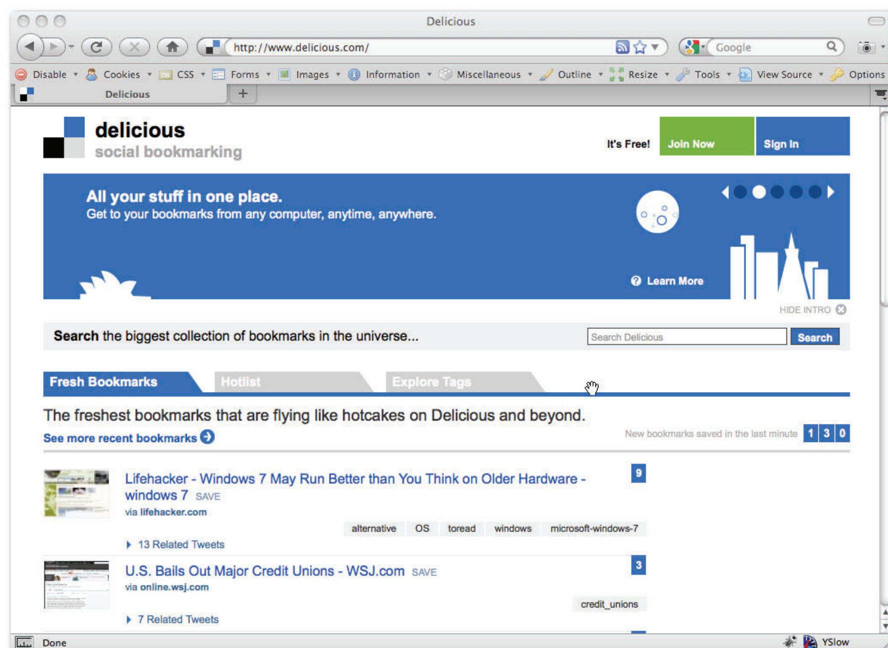


图9-16　著名社交书签站点Delicious

9

就像许多在线社交网站和应用一样，Delicious有着了不起的开发者社区和API（应用编程接口）。Delicious几乎对每个页面都支持RSS（简单内容聚合）订阅源（参见图9-17），包括最流行的书签页和每个人自己的书签页。RSS订阅源基于XML标准生成。

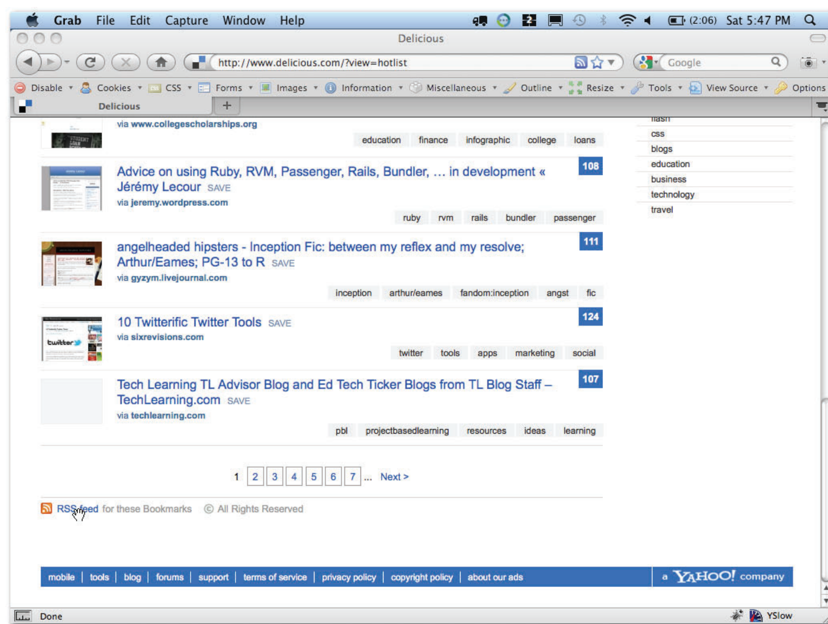


图9-17　Delicious RSS订阅源

Delicious也提供API接口，供那些希望在自己的网站和移动应用中使用Delicious API的高级用户使用（参见图9-18）。通过这一API，我们能够提交内容到Delicious，也能够（在自己的应用中）显示从Delicious获取的内容。要实现这些必须得到API key形式的授权。

在下面的教程中，我将使用Delicious订阅源。Delicious站点提供XML订阅源和JSON订阅源。我们可以使用查询字符串参数获取特定的内容，但无法通过订阅源添加任何内容到Delicious。我们可以参阅Delicious站点上的文档查看查询参数的完整列表，也可以查看图9-19。

使用我的用户名`jakerutter`作为URL中查询字符串的查询参数（http://feeds.delicious.com/v2/json/jakerutter?count=10），能够获取我添加到Delicious的最新10个书签（参见图9-20）。每位用户都有一个订阅源，我们可通过关注朋友的订阅源查看他们创建的所有书签。

在这个解决方案中，我们编写一个My Delicions Feed（我喜欢的书签）组件。可以把它放到自己的站点上，或者任何其他使用了jQuery的站点上（参见图9-21）。这个教程用到了JSONP（支持填充的JSON）技术，这是实现跨域Ajax调用必不可少的技术。JSONP通过把返回的JSON数据包装在你指定的回调函数中实现跨域调用。没有JSONP，跨域调用就会失败。我们无法使用那些不支持JSONP请求的API。

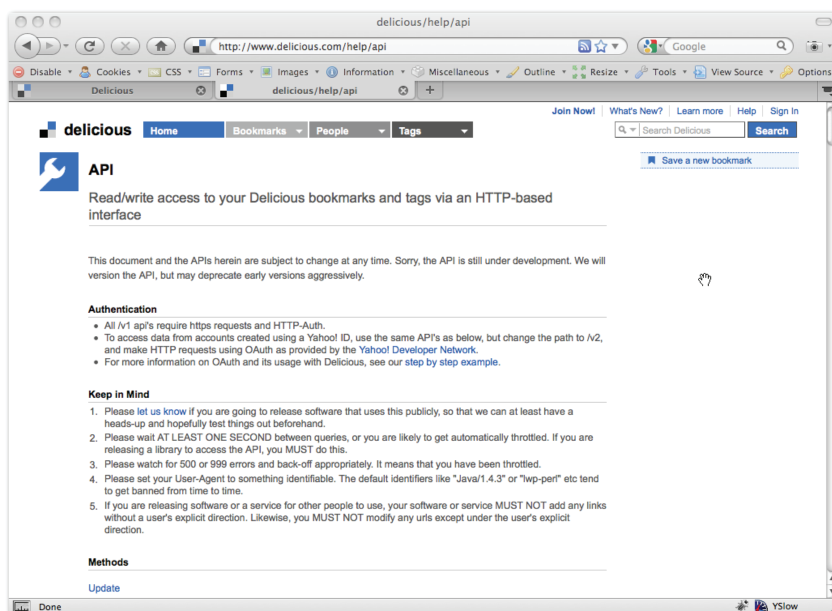


图9-18　Delicious API文档

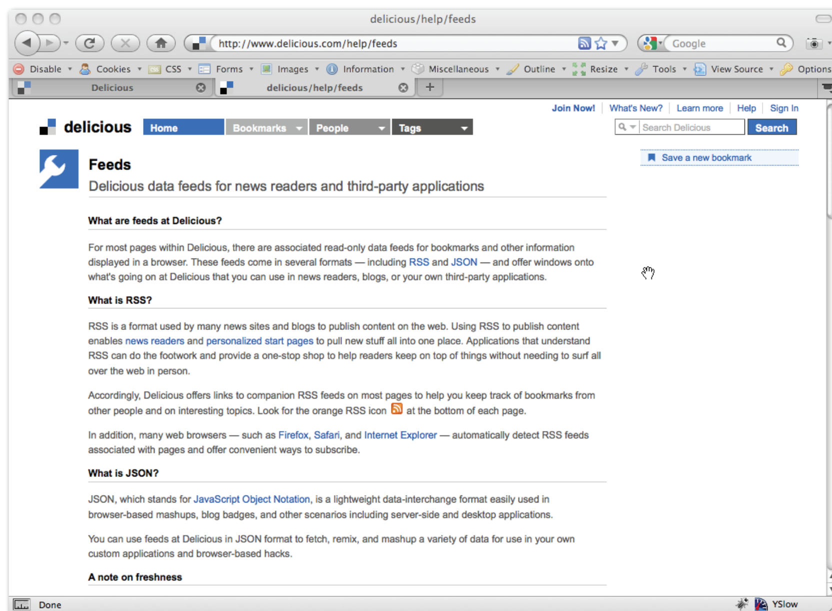


图9-19　Delicious 订阅源文档

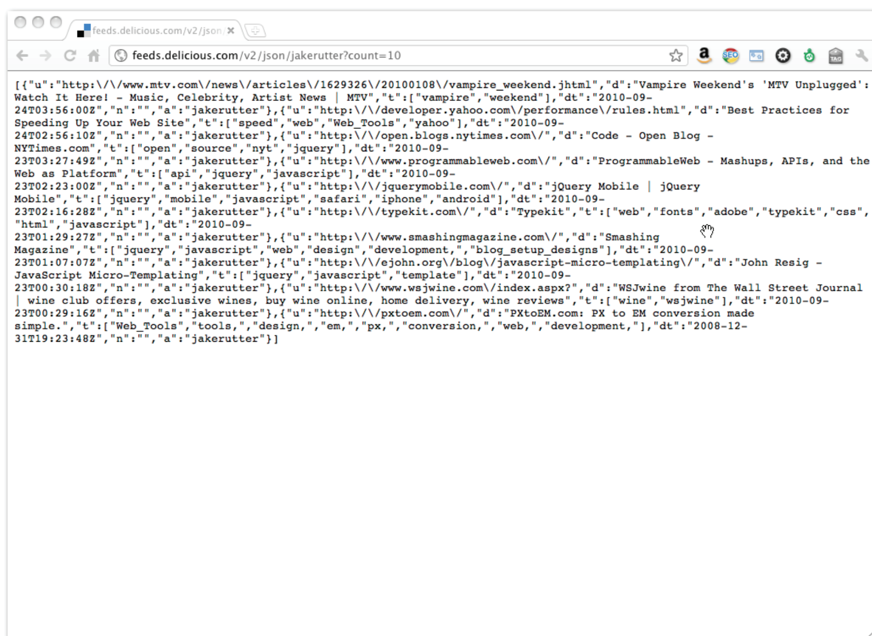


图9-20　我的书签JSON订阅源

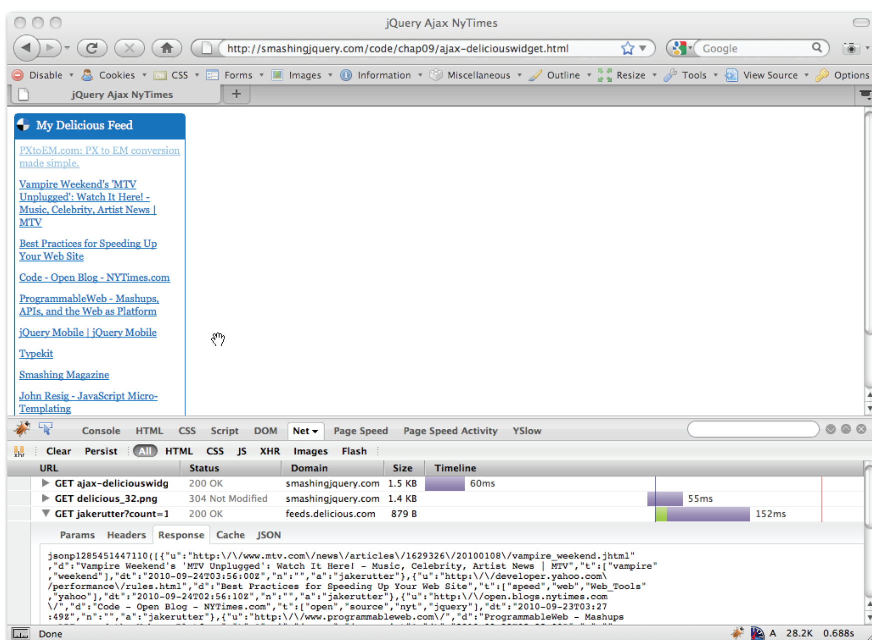


图9-21　动画Delicious书签组件

(1) 下面这些CSS控制组件在页面上的布局。我们可以在jQuery编程之前先写好所有的CSS，也可以先打造一个大概的基础，然后边写程序边改样式。下面的CSS示例代码包括一些Mozilla Firefox浏览器和Safari WebKit专用的圆角样式，其他浏览器只能通过border属性看到直角效果。

```
body{
   font-family:georgia;
   font-size:12px;
}
.widget{
   border:1px solid #1179CB;
   -moz-border-radius:5px;
   -webkit-border-radius:5px;
   width:200px;
}
.widget h1{
   font:14px georgia;
   padding:5px;
   color:#fff;
   background:#1179CB url(images/delicious_32.png) 2px 5px no-repeat;
   text-indent:20px;
   height:20px;
   margin:0;
}
#user-feed{
   list-style-type:none;
   margin:0;
   padding:0;}
   #user-feed li{
   padding:5px;
}
a{
   color:#1179CB;
}
```

(2) 准备包含书签无序列表的HTML组件代码。在其中添加一个无序列表ul#news-feed。

```
<div class="widget">
  <h1>My Delicious Feed</h1>
  <ul id="news-feed"></ul>
</div>
```

(3) 创建变量query，并把它的值设置为我的Delicious订阅源URL：

```
var query = "http://feeds.delicious.com/v2/json/jakerutter?count=10";
```

(4) 接着调用$.ajax()方法并提供4个参数：type参数使用GET；dataType参数是JSONP；url参数是变量query；回调函数是processData：

```
$.ajax({
  type: "GET",
  dataType: "jsonp",
  url: query,
  success: processData
});
```

9

(5) 编写一个空白的回调函数processData。它有一个参数data，代表收到的JSON数据。

```
function processData(data) {
}
```

(6) 使用$.each()方法创建一个循环，遍历data中的数据。i是数组的索引，item则是相应的值：

```
function processData(data) {
  $.each(data, function(i, item) {
  });
}
```

(7) 在$.each()方法内，使用$('
')动态生成一个li元素，然后使用.html()方法，把一个a标签纳入li元素，之后把它追加到ul#news-feed中去。在每个a标签 中，URL和标题均使用item和键（的名字）来访问相应的值（数据），在本例中.u是URL，.d是标题。

```
function processData(data) {
  $.each(data, function(i, item) {
    $('<li></li>').html("<a href='"+item.u+"'>"+item.d+"</a>").appendTo
     ('#user-feed');
  });
}
```

(8) 新建变量newsInterval，并将它的值设置为2000 ms，它的值是刷新My Delicious Feed组件的时间间隔。

```
var newsInterval = 2000;
```

(9) 接下来，我新建一个函数slideArticle()，它负责Delicious书签组件的所有特效。该函数的第一条语句选中ul#news-feed的最后一项，克隆它，然后使用.prependTo()方法将它添加到列表的最上方。第二条语句选中ul#news-feed的第一项，然后调用.slideDown()方法实现滑入效果。我想让这个元素在500 ms的时间内渐渐滑入，而且还希望它在1000 ms的时间内淡入。通过在同一条语句中链式调用.fadeIn()方法和.slideDown()方法，我实现了预期效果。slideArticle函数的最后一条语句用于移除列表的最后一项。这3条语句顺序执行，极佳地模拟了淡入及滑动效果。

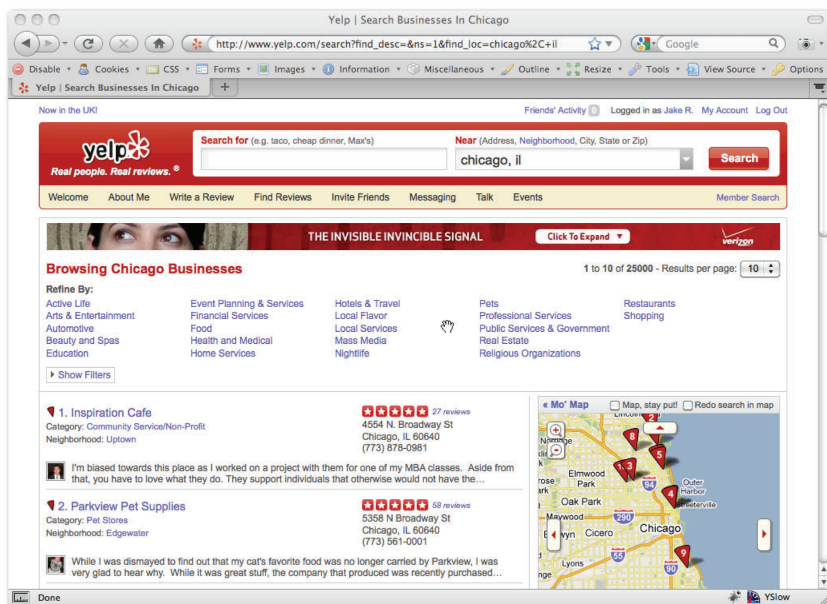
```
function slideArticle() {
    $('#user-feed li:last').clone().prependTo('#news-feed').css('display','none');
    $('#user-feed li:first').fadeIn(1000).slideDown(500);
    $('#user-feed li:last').fadeOut().remove();
}
```

(10) 最后一段JavaScript代码可以说是最重要的。我需要设置setInterval函数，以便每隔newsInterval（2000 ms）就执行slideArticle一次。这个函数一直循环不停。没有这个函数，Delicious书签组件就无法运行。

```
setInterval(slideArticle, newsInterval);
```

9.9 使用 JSONP 和 Yelp API 创建一个 Yelp 最热点评组件

Yelp（www.yelp.com）网站成立于2004年，提供类似“大众点评”[①]的本地搜索及点评服务。这一网站提供本地商务（从餐饮到书店，几乎无所不包）点评及评级（参见图9-22）。那些体验过这些企业服务水平的当地顾客们，不断在这个在线社区留下反馈信息，为该社区带来价值。到了2007年，Yelp改进了站点，为开发者提供API以便他们创建基于Yelp数据的应用程序，这极大地提高了Yelp点评的覆盖面。



由yelp.com许可复制

图9-22　Yelp上的芝加哥地区商业点评

Yelp文档站（www.yelp.com/developers/documentation）是专为开发者社区提供的文档站点，从这里可以链接到Yelp API、代码示例、API文档等（参见图9-23）。这个文档站结构合理，能保证开发者使用正确的工具使用这一公司提供的API编写自己的应用，值得其他提供API服务的公司借鉴。

① “大众点评”是国内的一家城市生活消费指南网站，网址为http://www. dianping. com。

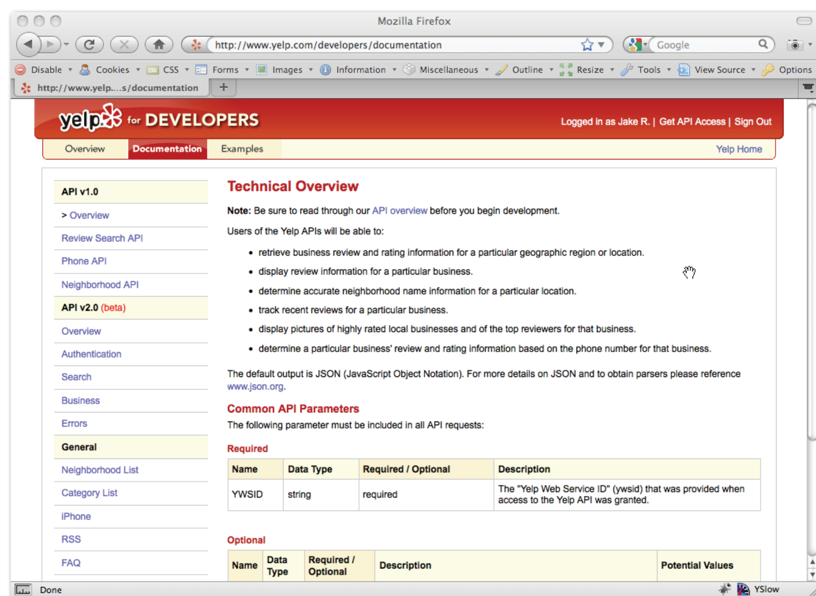


图9-23　Yelp文档站

9.9.1　申请Yelp API Key

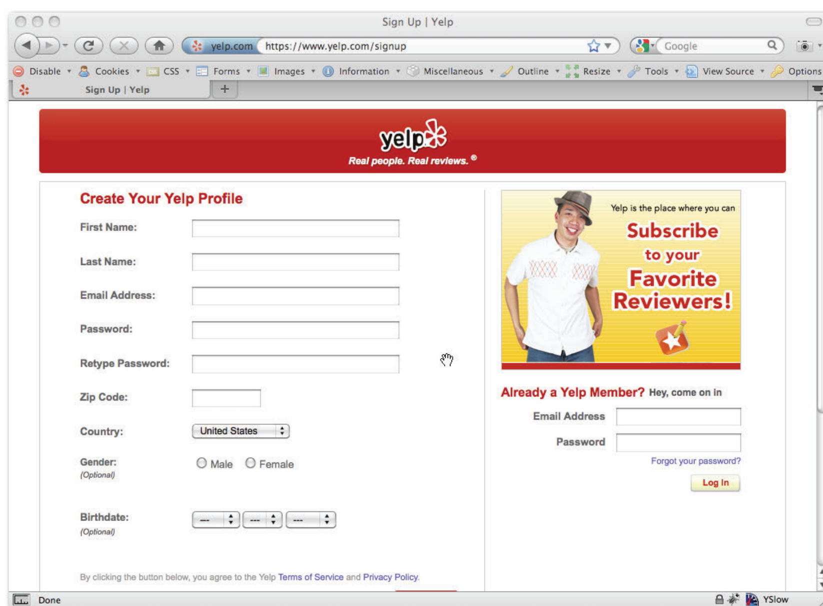
在使用任何第三方API之前，我们通常都需要先申请一个API key（某些情况，如我们在前面体验的Delicious API是个例外）。第三方站点通过API key来监控应用的活跃程度：通过这个API，第三方服务器能够记录我们应用的请求数和流量。绝大多数API至多支持每日100个请求，审批通过之后可以支持更多的请求及流量。使用这些API有时还会产生费用，在开始任何开发工作之前，一定要认真了解这些API。

对未得到批准的开发项目，Yelp API支持每天不超过100个请求。在项目正式上线并审批通过之后，Yelp放宽限制为每日不超过10 000个请求。Yelp API支持通过3种方法依据企业的类型、企业的位置以及附近的企业搜索企业，这3种方法是点评搜索、电话搜索及附近搜索。

审批过程包括提交一个描述应用用途以及应用如何使用Yelp API的简单表单。提交申请之后，你会收到一封来自Yelp的电子邮件，分配给你一个账号管理员，他负责审核你的应用并最终通过或者拒绝你的项目。当把Yelp API整合到项目之后，Yelp还会为你的项目通过Yelp的认证提供帮助。我将在下面的解决方案中讲解这个过程。

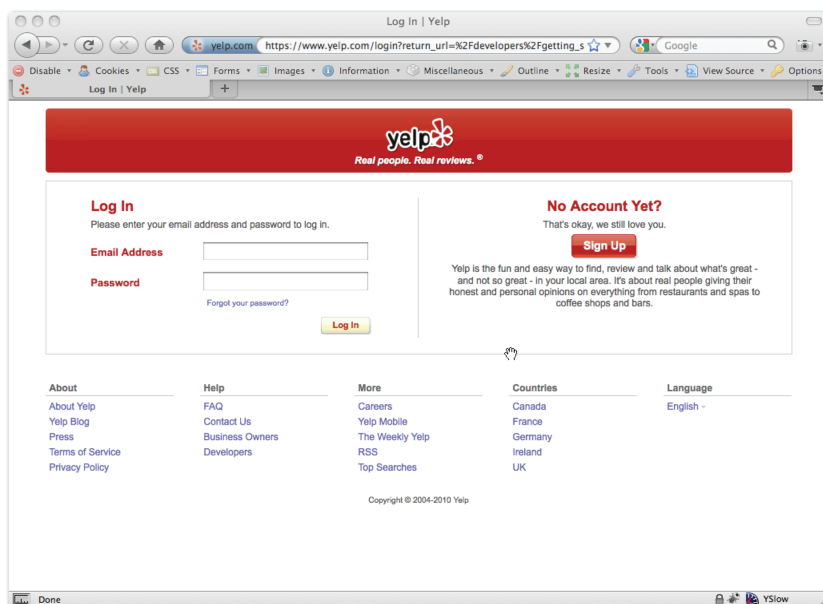
(1) 在申请Yelp API Key之前，需要先有一个Yelp普通账号，获取方法是访问www.yelp.com/ignup（参见图9-24）并提供特定信息（基本上就是出生日期、电子邮件地址等）。

(2) 有了一个Yelp账号之后，需要登录图9-25所示的Yelp开发者网站，或者访问www.yelp com/login?return_url=%2Fdevelopers%2Fgetting_started%2Fapi_access。



由yelp.com许可复制

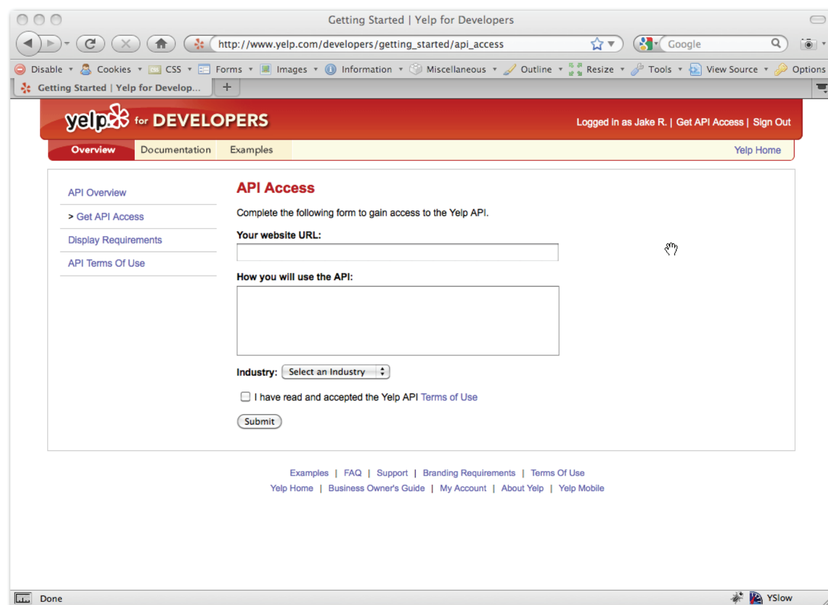
图9-24　Yelp注册账号页



由yelp.com许可复制

图9-25　Yelp开发者登录页

(3) 成功登录开发者网站之后，我们需要访问位于www.yelp.com/developers/getting_tarted/api_access 的API访问页面（即“API_Access”页面，参见图9-26）。



由yelp.com许可复制

图9-26　Yelp API申请访问权限页

在注册访问API时，需要填写自己的站点URL并描述将如何使用Yelp API。在得到访问API的授权之后，你会收到几封来自Yelp的邮件，指导你使用Yelp API等。当应用开发完成之后，需要在Yelp上展示你的应用给分配给你的账号管理员，请他审核并批准应用上线。他们通常会检查你的应用是否使用了Yelp的Logo以及是否清楚地显示了必要的信息（一些术语和约束条件，你在注册API申请时同意过的那些东西）。

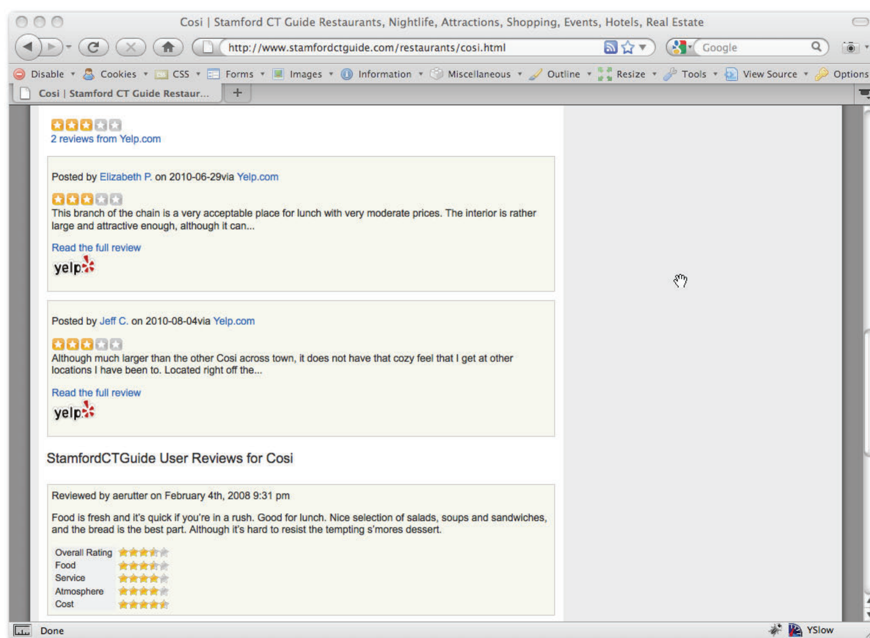
9.9.2　使用Yelp API基于电话号码获取点评

StamfordCTGuide.com是我创建并管理的一个Web一个项目。这是位于康涅狄格州Stamford的一家本地在线指南网站，有350多个页面，由我和妻子共同管理。

StamfordCTGuide.com站点是在WordPress（一个著名的博客及内容管理系统服务平台）上开发的，使用了一个支持用户点评餐馆的WordPress插件。我们这个小小的站点还不能吸引大量的点评，而Yelp上有着大量的信誉点评及评级内容。在通读过Yelp的文档后，我决定整合Yelp和StamfordCTGuide.com，以便为我站点上列出的每一家餐馆导入Yelp上的点评（参见图9-27）。

我在前面提到过，Yelp提供了不少API：Review Search（点评搜索）、Phone（电话搜索）及Neighborhood（附近搜索）。我对Review Search和Phone Search特别感兴趣。StamfordCTGuide.com

上列出的每一家餐馆都有街道地址及联系电话。我首先使用Review Search API并用URL http://pi.yelp.com/business_review_search?location=650%20Mission%20St%2ASan%20Francisco%2A%20CA&ywsid=XXXXXXXXXXXXXXXX 尝试传入餐馆地址数据给这个URL，以便得到点评列表。我遇到的一个问题是Review Search API并非总是返回一条结果，绝大多数情况下都会返回多条结果（参见图9-28）。



由yelp.com许可复制

图9-27 在StamfordCTGuide.com上整合了Yelp的内容

测试一番之后，我发现原来这是因为我提交给Yelp网站的地址不规范造成的。我提交的数据并非总是包括邮政编码和正确的街道地址，而Yelp API使用地址匹配点评。如果提供的并非完整地址，API就很难精确匹配。有些API调用返回一条结果，而其他的则返回5条。这条路看来走不通。

还剩下一个选择，那就是使用Phone Search API来尝试匹配那些同时存在于我的站点和Yelp站点的那些餐馆。查阅过我的餐馆数据库之后，我发现登记有电话号码的餐馆数量远大于登记有完整地址的餐馆。使用Yelp的Phone Search应该能得到更一致的的数据匹配。

在动手编码之前，我尝试着使用几个本地餐馆的电话号码发出请求，发现这一API仅返回这些电话号码对应的餐馆列表（参见图9-29）。如果电话号码不匹配，就什么也不返回。举个例子，我们可用这个URL结构请求Phone Search API：http://api.yelp.com/phone_search? phone=1234567890&ywsid=XXXXXXXXXXXXXXXX。

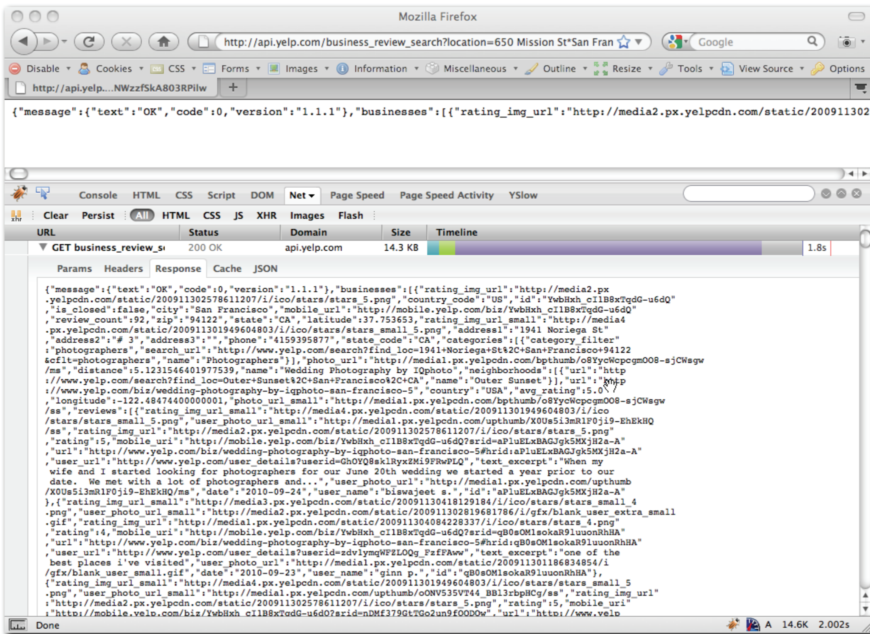


图9-28　点评搜索结果

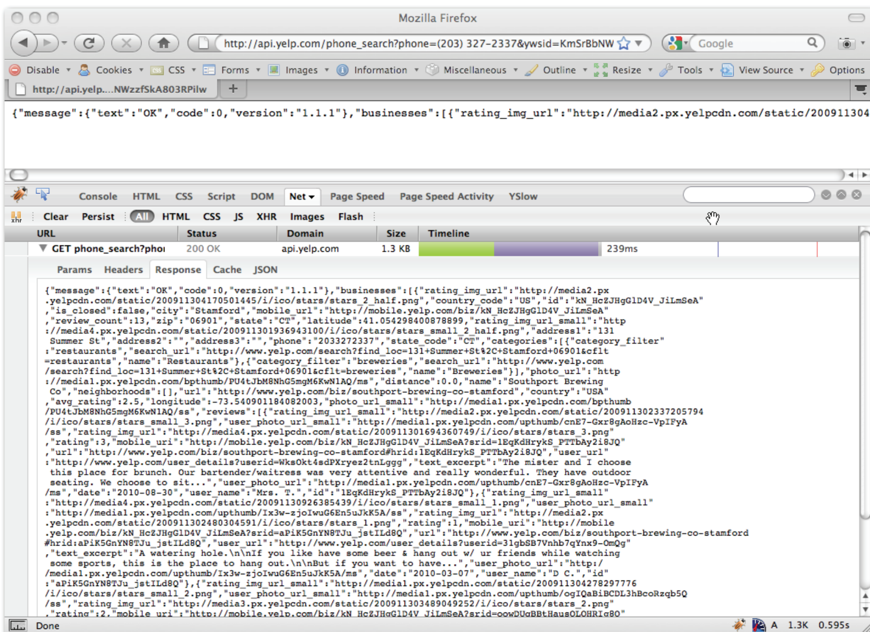


图9-29　电话搜索结果

在研究了如何更好地把Yelp点评及评级API整合到stamfordCTGuide之后，我开始转入下一步——考虑具体实现：使用jQuery获取Yelp点评，并把它们导入我的站点。

由于Yelp API与StamfordCTGuide.com域名不同，整合只能使用跨域请求，我需要编写使用JSONP的回调函数，但这又引出一个问题：Yelp API v1.0尚不支持JSONP，不过我们可以解决这个问题，但替代办法只支持获取JSON数据。我们不能够添加点评、评级等，不过没关系，因为我实现的这个版本尚不需要这些功能。

替代办法需要使用API请求链接在网页源代码中生成一个script标签，该标签会立刻被求值执行。使用这项技术解决跨域问题纯粹是因为我只需要从Yelp获取数据，这是一种单向通信。在页面加载成功之后，就再不会调用API。我们仅在页面初始化时调用API从Yelp网站载入数据。

这个解决方案需要综合使用HTML、CSS、jQuery和PHP。我使用PHP获取特定地址的电话号码并把它赋值给一个JavaScript变量locPhone。

(1) 在页面上添加以下HTML，用它存放来自Yelp的点评数据。创建一个div并把它的ID设置为yelpReviews。

```
<div id="yelpReviews" style="margin:5px;"></div>
```

(2) 在document的ready事件处理函数中，添加一行调用writeScriptTag函数的语句。我们将在下一步中定义这个函数。我们给该函数传一个URL参数，参数中包含电话号码、API Key、返回结果数限制以及回调函数的名字。

```
$(document).ready(function(){
  writeScriptTag( "http://api.yelp.com/phone_search?"+
  "&phone="+locPhone+
  "&ywsid=KmSrBbNWzzfSkA803RPilw"+
  "&limit=1"+
  "&callback=showData"); // <- callback
});
```

(3) 接下来实现writeScriptTag函数。在该函数内，创建一个变量yelpScript，我们使用它在页面加载完成之后生成script标签。要确保标签的type值为text/javascript，并且src属性设置为传入的path参数。最后一条语句把这个script标签追加到document的body标签中。

```
function writeScriptTag(path) {
 var yelpScript=document.createElement('script');
  yelpScript.type='text/javascript';
  yelpScript.src=path;
  $("body").append(yelpScript);
}
```

(4) 写一个空的回调函数showData，它接受一个data参数（JSON对象）。

```
function showData(data) {
}
```

(5) 我们需要使用$.each()方法创建两个循环。第一个循环遍历data.businesses对象，数组索引是i，对应的值是business。第二个循环嵌套在第一个循环内，遍历business.reviews，数组的索引是i，对应的值是review。

```
function showData(data) {
  $.each(data.businesses, function(i,business){
    $.each(business.reviews, function(i,review){
    });
  });
}
```

(6) 在第一个循环中，我们可得到对企业的点评总数并把它插入到页面中的#yelpReviews元素中。这也是Yelp网站的要求，应用总是显示点评总数并提供到Yelp网站上商家页面的链接。图9-29展示的是通过Phone Search API获取的所有Yelp数据。

```
function showData(data) {
  $.each(data.businesses, function(i,business){
  var bizContent = '<p><img src="' + business.rating_img_url + '" alt=""/><br>
    <a href="'+ business.url +'">'+ business.review_count + ' reviews from
Yelp.com</a></p>';
  $(bizContent).appendTo('#yelpReviews');

    $.each(business.reviews, function(i,review){
  });
});
}
```

图9-30展示了Yelp网站上的响应值。

Mozilla Firefox

http://www.yelp.com/developers/documentation/phone_api

Category List
iPhone
RSS
FAQ
Support

Response Values:

This section outlines the standard response values from a "BusinessReviewSearch" operation.

Node Element	Type	Definition	Max Number In Result Set
businesses	list	node that contains a list of business entries	1
address1	node	line one of business address	1 per business
address2	node	line two of business address	1 per business
address3	node	line three of business address	1 per business
categories	list	provides a list of categories that this business is associated with	1 per business
categories.name	node	name of one category to which this business is associated with	unbounded
categories.search_url	node	URL to execute a search for this particular category on Yelp.com	unbounded
city	node	name of city that business is located in	1 per business
distance	node	distance that business is from center point of city	1 per business
id	node	Yelp Id for this business	1 per business
latitude	node	latitude of business	1 per business
longitude	node	longitude of business	1 per business
mobile_url	node	URL for mobile business page on Yelp	1 per business
name	node	name of business	1 per business
nearby_url	node	URL to execute search for nearby businesses	1 per business
neighborhoods	list	List that provides neighborhood(s) information for business	1 per business

由yelp.com许可复制

图9-30　Yelp网站上的响应值一览表

(7) 在第二个循环中，我们将通过review对象访问每一条点评。我们可得到每一个用户、他们的点评URL、点评日期、评级图片并利用索引得到每一条评级摘要。Yelp也要求在每一条Yelp点评旁边显示Yelp Logo，以说明来源。每条点评都插到#yelpReviews元素之后，分别包含在它们自己的.comments-block元素中。

```
function showData(data) {

  $.each(data.businesses, function(i,business){
  // 附加的循环
  var bizContent = '<p><img src="' + business.rating_img_url + '" img=""/><br>
    <a href="'+ business.url +'">'+ business.review_count + ' reviews from
    Yelp.com</a></p>';
  $(bizContent).appendTo('#yelpAVG');

    $.each(business.reviews, function(i,review){
    var content = '<div class="comments-block"><p>Posted by <a href="'
      review.user_url+'">' +review.user_name + ' </a> on ' + review.date
      + 'via <a href="'+review.url+'">Yelp.com</a>';
    content += '<p><img src="' + review.rating_img_url + '" img=""/><br>';
    content += review.text_excerpt  + '</p>';
    content += '<p><a href="'+review.url + '">Read the full review</a><br>';
    content += '<img src="http://media3.px.yelpcdn.com/static/200911302846157596/
       /developers/yelp_logo_50x25.png"></div>';
    $(content).insertAfter('#yelpReviews');
    });
  });
}
```

这个实现真是既简单又有趣！不妨想象一下，一定会有越来越多的公司为他们网站数据提供API。在那些支持JSON-P请求的站点上，我们可以使用这些API并轻松使用来自其他网站的内容，却根本不必要编写大量的后端代码，这一切都归功于jQuery。

第 10 章

10

创建及使用jQuery插件

除了可以自己编写功能强大的插件扩展jQuery的核心功能外，Web设计师和开发者还可以在站点或应用程序中使用来自开源社区的插件。任何Web设计师或开发者都能够编写jQuery插件，但编写插件需要具备中高级JavaScript知识。

本章，我会带你了解如何安装流行的jQuery插件，如jQuery UI、jQuery Tools和Fancybox（lightbox的替代品）。在本章的末尾，我会介绍如何编写及发布jQuery自定义插件。

10.1 jQuery 插件

jQuery插件有着广泛的应用。举例来说，网站上的绝大多数lightbox窗口（lightbox window）都是由jQuery插件完成的。在不离开当前页面的前提下，lightbox窗口可以用来显示较大的图像，比如在一个产品目录站点中放大显示一个图片。只要在页面中放一个包着大图的div层，然后将它显示在半透明的背景上，就能实现这一功能。著名的博客平台WordPress（www.wordpress.com）在其管理界面使用了一个lightbox窗口类型的插件，通过它给一篇博客文章添加媒体文件（参见图10-1）。

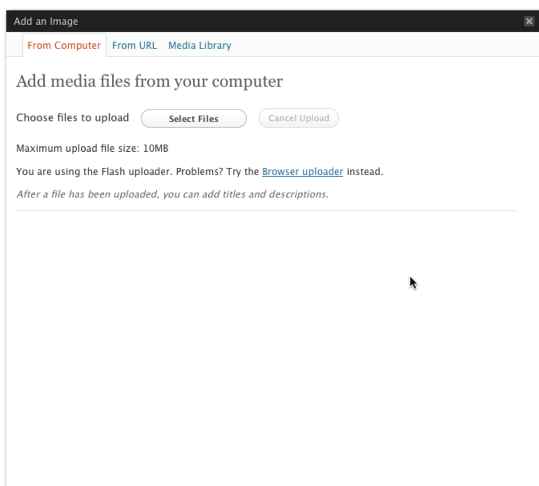


图10-1 WordPress在管理界面使用一个lightbox窗口插件为博文添加图片文件

jQuery有着庞大的插件开发者社区，他们创建了图库、表单处理及表单元素处理等被广泛应用的伟大插件。编写jQuery插件需要JavaScript高级知识，因为你的脚本必须能够部署、应用于很多不同的场合。

下面是使用插件的一些小提示。

- 检查该插件是否与你正在使用的jQuery版本兼容。
- 检查插件是否兼容各种浏览器（IE、Firefox、Safari、Chrome和Opera）。
- 检查插件的发布日期，如果该插件发布于好几年前，则意味着这个插件很可能已被废弃了。
- 阅读插件所带文档，看看该插件的实现是否容易，以及该插件是否支持进一步扩展。
- 检查社区对该插件的反应。是否已经有很多人成功地使用了该插件？
- 在使用插件过程中若遇到问题，而该插件来自jQuery插件站点（http://plugins.jquery.com）或者Google Code（http://code.google.com/hosting），那么你最好提交一个bug报告，这样开发者就能及时知道这个问题。
- 如果插件已经明显过时（比如发布于很久以前），最好找找有没有新版本。

10.2　在站点上使用 jQuery 插件

在站点上使用jQuery插件是一件相当简单的事。回想一下，你是怎么将jQuery库包含到自己站点中的？你只需要像包含jQuery库一样把所要用的插件包含到页面中。记着插件文件的包含位置应该在jQuery库之后，否则它就不能工作。

只要你愿意，想包含多少个插件都行。包含很多个单独插件唯一的缺点是在加载页面时会增加对Web服务器的请求，这可能会导致页面加载时间较长，响应缓慢。图10-2展示了Firebug中一个页面加载多个jQuery插件的细节信息。一个避免多次请求的简单方法是将这些插件合并到一个文件（比如叫jquery.plugins.js）中。这样，只包含这一个文件就够了。

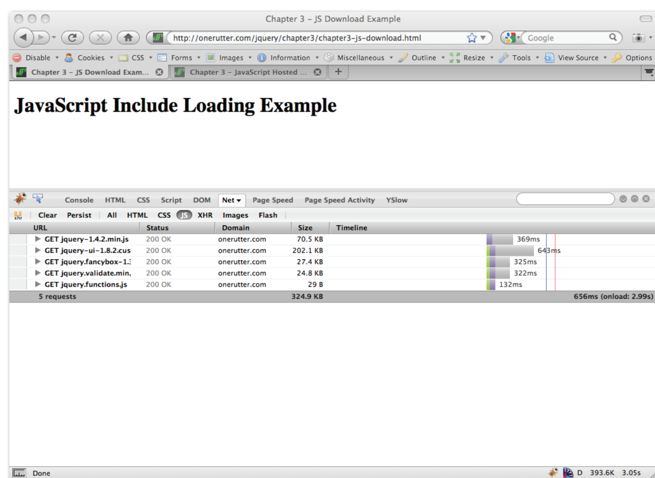


图10-2　Firebug显示一个页面中包含多个jQuery插件

10.3　在站点上包含 jQuery UI

jQuery有一个官方的用户界面库jQuery UI，见图10-3，其官方网站是 jqueryui.com。jQuery UI库独立于jQuery库并依赖jQuery库。



图10-3　jQuery UI 首页

jQuery UI库主要由人机交互组件和一些小部件组成。交互组件是一些能帮助网站应用完成特定人机交互任务的脚本，比如Droppable组件的主要功能是为应用程序提供拖放支持。

表10-1概要列出了jQuery UI库的所有功能组件，本章后续几节将点评jQuery UI库中最常用的几个组件。

表10-1　jQuery UI 组件

交互组件	小　部　件	实用功能	特　　效
Draggable	Accordion	Position	Show
Droppable	Autocomplete		Hide
Resizable	Button		Toggle
Selectable	Date Picker		
Sortable	Dialog		
	Progress Bar (forms)		
	Slider (forms)		
	Tabs (forms)		

小部件是一些自给自足的部件。拿Accordion（折叠菜单）部件来说，把它直接嵌入到应用程序中，就能为用户界面提供交互效果。在这儿，我会重点讲讲如何在站点中应用这些小部件。

jQuery UI还提供了一些方法扩展jQuery核心事件，比如在事件中动态地改变背景或边界颜色。jQuery Easing插件现在已经是jQuery UI核心的一部分，这使得每个人都能为他们已有的动画增添更强大的缓动效果。

jQuery UI主要用于因特网富客户端应用程序（RIA），在这些程序中，人与浏览器之间会进行大量的交互，所有的jQuery UI部件都有用武之地。

10.3.1 下载jQuery UI

我们可以在 http://jqueryui.com/download 下载jQuery UI库（图10-4）。你可以直接下载整个UI库，也可以使用Build Your Download 页（自定义下载页）仅下载自己用到的那部分组件。这个选项非常有用，因为jQuery UI库比jQuery大很多，我们在程序中很少会用到库里所有的组件。

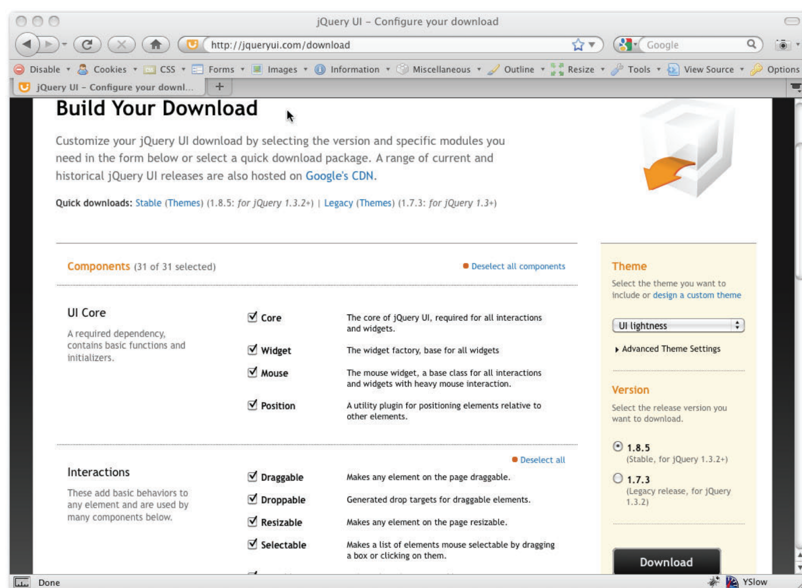


图10-4 jQuery UI自定义下载页

10.3.2 将jQuery UI添加到站点

类似于jQuery库，jQuery UI作为一个单独的JavaScript文件，也要上传到Web服务器，然后在页面头部jQuery库之后加载。另外，Google CDN也提供jQuery UI下载服务。

```
<script src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.5/jquery-ui.
  min.js"></script>
```

通过Google CDN包含jQuery UI库的好处是有更好的性能和文件缓存，缺点是你不得不包含整个jQuery UI库，即使其中有很大一部分组件你根本不需要。

最终包含jQuery UI和jQuery库的页面头部会像下面这样：

```
<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>
<script type="text/javascript" src="js/jquery-ui-1.8.4.custom.min.js"></script>
```

10.3.3　jQuery UI小部件工作原理

每个小部件都拥有丰富的API，包括选项、事件和方法。利用这些API，我们可以在站点上创建自定义组件，并轻松把它用在别的地方。图10-5展示了Tabs小部件页。Tabs组件要么通过默认设置，要么通过API选项完成功能。

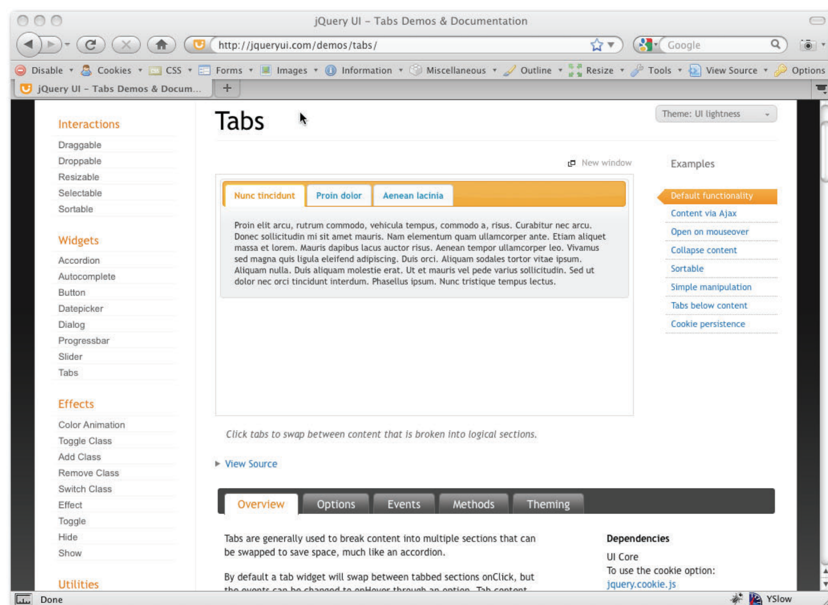


图10-5　jQuery Tabs小部件及所有文档

10.3.4　自定义jQuery UI的外观

jQuery UI库基于一个CSS框架，这意味着我们能够轻易地通过自定义实现自己的风格，或者使用ThemeRoller创建一个主题。10.3.5节将详细介绍ThemeRoller。一个主题就是适用于jQuery UI库中所有组件的CSS样式和图片的集合。通过jQuery UI CSS框架，我们能确保所有设计不仅适用于当前UI组件，而且适用于将来的UI组件。

CSS框架的优雅之处在于所有组件均使用相同样式，如果你为某种活跃状态设置了样式，则

该样式自动应用于所有的组件。

该CSS框架由两个样式表文件组成：

- ui.core.css
- ui.theme.css

下面是一段由一个jQuery UI组件渲染出来的HTML代码片段:

```
<div role="tablist" class="ui-accordion ui-widget ui-helper-reset
ui-accordion-icons" id="accordion">
<h3 tabindex="0" aria-expanded="true" role="tab" class="ui-accordion-header
ui-helper-reset ui-state-default ui-state-active ui-corner-top">
<span class="ui-icon ui-icon-triangle-1-s">
</span>
<a tabindex="-1" href="#">
NY Mets
</a>
</h3>
<div role="tabpanel" style="height: 93px;" class="ui-accordion-content
ui-helper-reset ui-widget-content ui-corner-bottom ui-accordion-content-active">
<p>
The New York Mets are a professional baseball team based in the borough of Queens in
New York City. The Mets are a member of the East Division of Major League Baseball's
National League. The Mets are also often referred to as the "Amazins" by fan and media
alike.
</p>
<p>One of baseball's first expansion teams in 1962, the Mets won the 1969 World Series.
They have played in a total of four World Series, the most of any MLB expansion team,
including a second dramatic win in 1986.</p>
</div>
<h3 tabindex="-1" aria-expanded="false" role="tab" class="ui-accordion-header
ui-helper-reset ui-state-default ui-corner-all">
<span class="ui-icon ui-icon-triangle-1-e">
</span>
<a tabindex="-1" href="#">
NY Jets
</a>
</h3>
<div role="tabpanel" style="height: 93px; display: none;" class="ui-accordion-content
ui-helper-reset ui-widget-content ui-corner-bottom">
<p>
The New York Jets are a professional Football team based in East Rutherford, New Jersey,
representing the New York metropolitan area. They are members of the Eastern Division
of the American Football Conference (AFC) in the National Football League (NFL). In
a unique arrangement, the Jets share New Meadowlands Stadium (located in East Rutherford,
New Jersey) with the New York Giants.
</p>
</div>
<h3 tabindex="-1" aria-expanded="false" role="tab" class="ui-accordion-header
ui-helper-reset ui-state-default ui-corner-all">
<span class="ui-icon ui-icon-triangle-1-e">
</span>
<a tabindex="-1" href="#">
Manchester United
```

```
</a>
</h3>
<div role="tabpanel" style="height: 93px; display: none;" class="ui-accordion-content
ui-helper-reset ui-widget-content ui-corner-bottom">
<p>
Manchester United Football Club is an English professional football club, based in Old
Trafford, Greater Manchester, that plays in the Premier League. Founded as Newton Heath
LYR Football Club in 1878, the club changed its name to Manchester United in 1902 and
moved to Old Trafford in 1910.
</p>
</div>
</div>
```

当需要调整组件的外观时，只需要修改ui.theme.css样式表。还有一个选择，也就是使用ThemeRoller（一个由jQuery UI团队创建的在线应用）设置自己的主题。当我们改变样式时，新样式的效果会立刻显示在由所有jQuery组件组成的一个演示页面上。

10.3.5　使用ThemeRoller创建UI主题

ThemeRoller(http://jqueryui.com/themeroller/)提供了一个在线向导，这使得设将其应定主题的字体、颜色、边界样式变得极为简单，可以从网上下载主题并为其应用这些选项，然后将其应用到你所有的项目中（参见图10-6）。如果懒得从零开始创建一个主题，ThemeRoller提供了24种主题供你选用和调整。

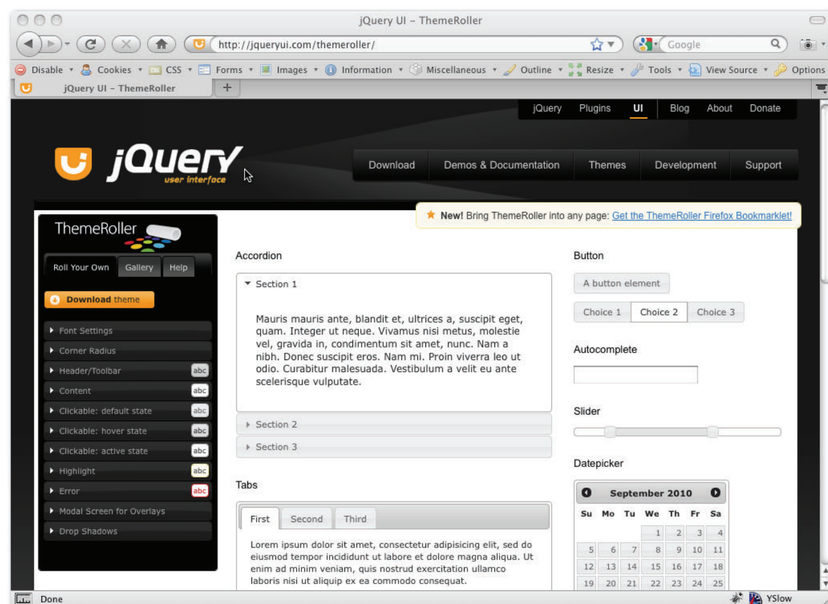


图10-6　jQuery UI ThemeRoller

(1) 设置ThemeRoller中的种种选项。创建一个主题时，我们可设置主题的以下样式：

- ❑ 字体设置；
- ❑ 圆角；
- ❑ 头部（Header）或工具栏（toolbar）；
- ❑ 内容；
- ❑ 可单击区域——激活状态、悬停状态和默认状态；
- ❑ 突出显示；
- ❑ 错误信息；
- ❑ 覆盖层的模态窗口；
- ❑ 阴影。

(2) 在ThemeRoller中的左侧是各设置项，通过表单设置样式。当你改变一个值，这个改变的效果就立即应用在右边所有的jQuery UI组件示例上。你可以随心所欲地对显示效果进行调整。

(3) 只需要单击Download Theme（下载主题）按钮，你就可以将刚刚创建好的主题下载到自己的机器上（参见图10-7）。

单击Download Theme 按钮

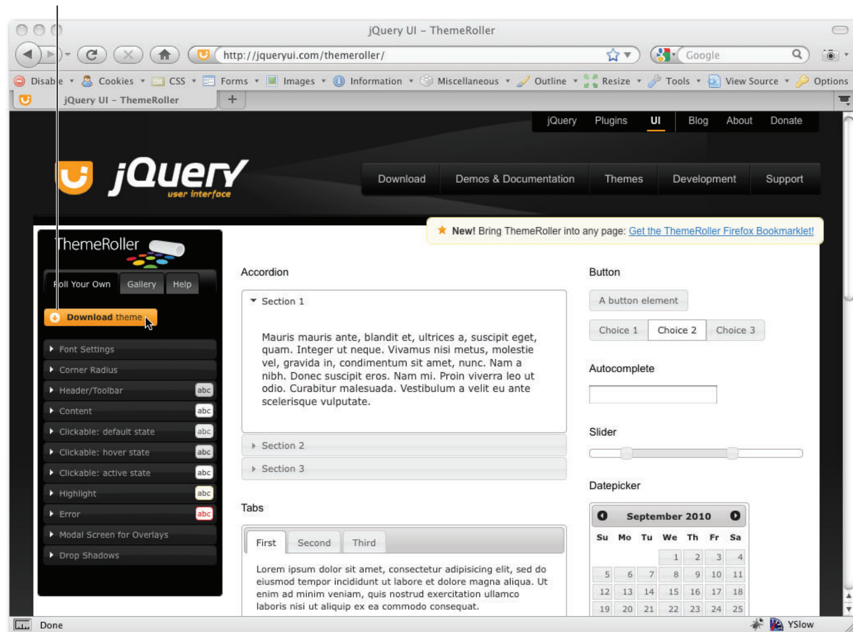


图10-7　ThemeRoller应用的Download Theme按钮

(4) 可以单独下载某个主题，也可将其打包到jQuery UI下载包中下载（参见图10-8）。只需反选所有的jQuery UI组件，然后只下载这个主题。

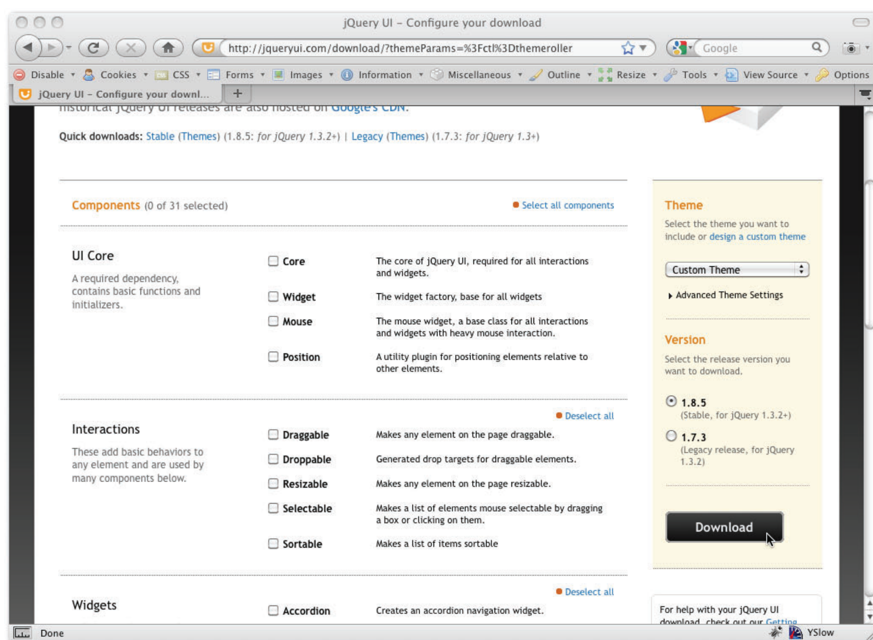


图10-8　反选了所有jQuery UI组件的Build Your Download页

(5) 主题被打包成一个扩展名为.zip的压缩文件，它的目录结构如下：

- /css/custom-theme/jquery-ui-1.8.5.custom.css
- /css/custom-theme/images/
- /development-bundle/
- /js/

其中主题有关的文件位于/css目录下。如果你已经下载过jQuery UI的JavaScript文件，完全可以忽略其中的/js目录。

10.3.6　使用jQuery UI主题

不管是通过ThemeRoller自定义一个主题，还是从零开始创建一个主题，我们都需要在用到它的网页的头部将它包进来，就像下面这样：

```
<link type="text/css" href="../themes/base/jquery.ui.all.css" rel="stylesheet" />
```

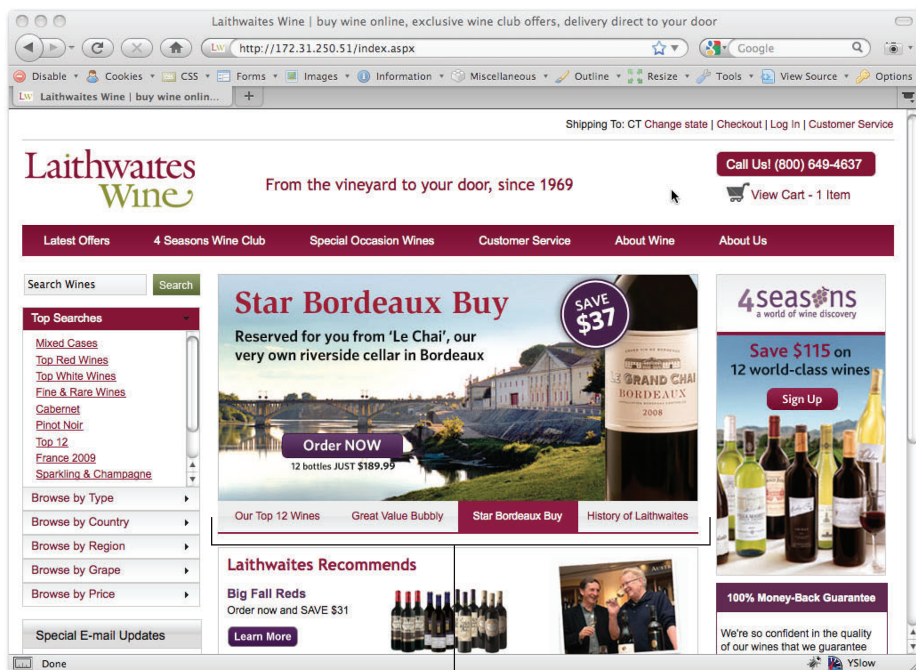
10.3.7　将jQuery UI组件整合到站点

在了解了jQuery UI插件的工作原理之后，我们就可以轻松地将所需的多个组件整合到站点。接下来，我将介绍如何设置tabs、Accordion、Autocomplete和Datepicker组件。

1. Tabs组件

jQuery UI tabs是一个小部件，当将它应用到DOM元素时，能动态地创建多个标签，为站点添加多种交互手段。我最近一次使用它，是为一个站点的首页创建可交互的幻灯片（参见图10-9）。

使用jQuery UI完成工作是如此简单，它（如此可靠）为我节省了大量的排错时间。例子中的tabs组件位于图片的下部。每个标签对应一张图片。切换标签，图片会相应的切换，并且应用淡入淡出的效果。



一种标签式幻灯片

图片截取自Laithwaiteswine.com

图10-9 Laithwaites Wine站点上的标签式幻灯片

我将在下面的教程中教你设置支持动画的标签式幻灯片，就像Laithwaites Wine网站上的一样。

(1) 通过Build Your Download页（http://jqueryui.com/download）下载包含tabs组件的jQuery UI包，然后把jQuery UI包含到页首部分：

```
<link type="text/css" href="../themes/base/jquery.ui.all.css" rel="stylesheet" />
```

(2) 包含自定义主题的CSS文件：

```
<link type="text/css" href="../themes/base/jquery.ui.all.css" rel="stylesheet" />
```

10

(3) 编写HTML。当我们使用jQuery UI组件时，有关的HTML代码必须遵守组件的约定，否则组件将无法正确地渲染这些HTML。将所有的HTML代码包到一个id="tabs"的div元素中，tabs组件将依据该id引用这段HTML。每个标签的id必须以tab开始，后面跟着一个连字符和顺序增长的序号（#tab-1、#tab-2、#tab-3……）。

```
<div id="tabs">
  <div id="tabs-1">
  <a href="#"> <img src="../images/homepage/4s_tab_offer1_july2010.jpg"
  alt="History of Laithwaites Wine" alt=""/></a>
  </div>
  <div id="tabs-2">
  <a href="#"><img src="/images/homepage/4s_tab_offer2_sep2010.jpg" alt="Kings
  of Cabernet" alt=""/></a>
  </div>
  <div id="tabs-3">
  <a href="#"><img src="/images/homepage/4s_tab_offer3_sep2010.jpg" alt=
  "Discover Chile" alt=""/></a>
  </div>
  <div id="tabs-4">
  <a href="#"><img src="/images/homepage/4s_tab_offer4_sep2010.jpg"  alt=
  "World-class Ros¨¦" alt=""/></a>
  </div>
  <ul>
    <li><a href="#tabs-1">History of Laithwaites </a></li>
    <li><a href="#tabs-2">Kings of Cabernet </a></li>
    <li><a href="#tabs-3">Discover Chile </a></li>
    <li><a href="#tabs-4">World-class Rose</a></li>
  </ul>
</div>
```

(4) 用jQuery选中#tabs元素，然后对其应用tabs方法。为了让标签能够自动轮播，设置tabs的rotate属性为7000，这样每隔7 s就会自动切换标签。

```
$("#tabs").tabs({ fx: { opacity: 'toggle' } }).tabs('rotate', 7000);
```

2. 折叠组件

折叠组件可用于呈现内容,也可用于在一个局促的空间里设置导航工具,在网上被广泛应用。当将它用于导航时,折叠组件把许许多多的菜单选项缩略为一个小菜单,让用户方便地漫游站点。有时，折叠组件有多种多样的折页（就像一页页幻灯片）并且内容可由Ajax动态生成，这样无须刷新页面就能得到新的内容。

在第6章中，我介绍了如何创建自己的折叠菜单或折叠内容，大约用了10行代码，而使用jQuery UI实现同样的效果，只需要一行代码（使用默认设置），见图10-10。真是简单到极点！

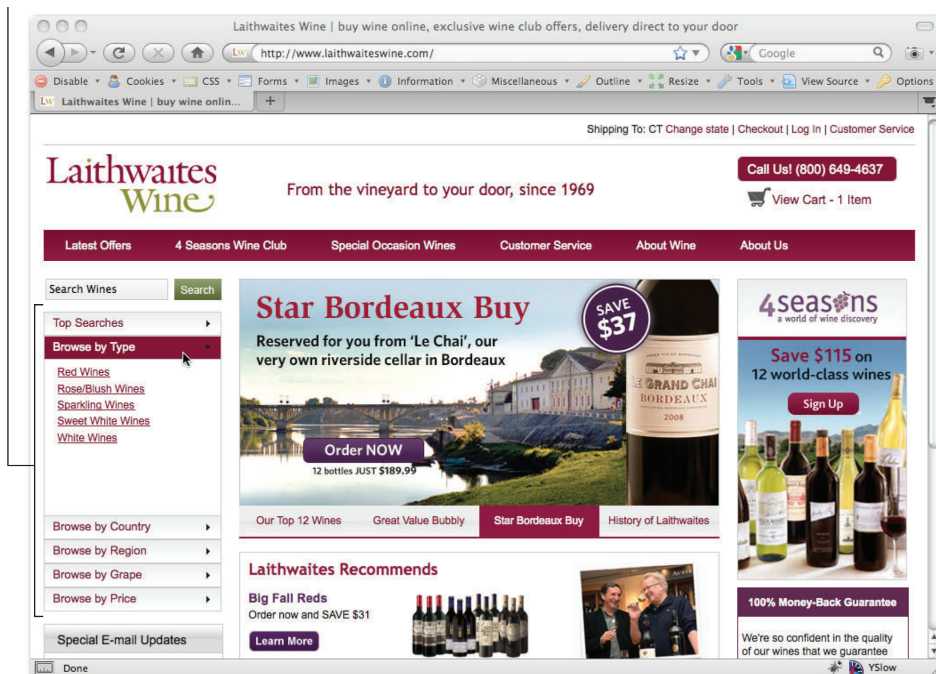
(1) 通过Build Your Download页（http://jqueryui.com/download）下载包含Accordion组件的jQuery UI包，然后把jQuery UI包含到页首部分：

```
<script type="text/javascript" src="js/jquery-ui-1.8.4.custom.min.js"></script>
```

(2) 包含自定义主题的CSS文件：

```
<link type="text/css" href="../themes/base/jquery.ui.all.css" rel="stylesheet" />
```

一个折叠菜单



图片截取自Laithwaiteswine.com

图10-10　Laithwaites Wine站点上的一个折叠菜单

(3) 编写HTML。当我们使用jQuery UI组件时，有关的HTML代码必须遵守组件的约定，否则组件将无法正确地渲染这些HTML。每个jQuery UI组件都有一个专门的文档网页，当你需要设置一个组件时，一定要参考这个文档网页。Accordion组件的文档页位于http://jqueryui.com/demos/accordion/。将所有折叠内容HTML包到一个`id="accordion"`的`div`元素中，Accordion组件将依据该`id`引用这段HTML。每块折叠内容都需要有一个`h3`元素（存放标题）和一个紧跟其后的`div`元素（存放内容）。

```
<div id="accordion">
  <h3><a href="#">NY Mets</a></h3>
  <div><p>The New York Mets are a professional baseball team based in the borough of
  Queens in New York City. The Mets are a member of the East Division of Major League
  Baseball's National League. The Mets are also often referred to as the "Amazins"
  by fan and media alike.</p>
  One of baseball's first expansion teams in 1962, the Mets won the 1969 World Series.
  They have played in a total of four World Series, the most of any MLB expansion team,
  including a second dramatic win in 1986.</p>
  </div>
   <h3><a href="#">NY Jets</a></h3>
   <div><p>The New York Jets are a professional Football team based in East Rutherford,
  New Jersey, representing the New York metropolitan area. They are members of the
  Eastern Division of the American Football Conference (AFC) in the National Football
```

```
League (NFL). In a unique arrangement, the Jets share New Meadowlands Stadium (located
in East Rutherford, New Jersey) with the New York Giants.</p>
  </div>
  <h3><a href="#">Manchester United</a></h3>
  <div><p>Manchester United Football Club is an English professional football club,
based in Old Trafford, Greater Manchester, that plays in the Premier League. Founded
as Newton Heath LYR Football Club in 1878, the club changed its name to Manchester
United in 1902 and moved to Old Trafford in 1910.</p>
  </div>
</div>
```

(4) 用jQuery选中`#accordion`元素，然后调用`.accordion()`方法。这个教程的显示效果见图10-11。

```
$("#accordion").accordion();
```

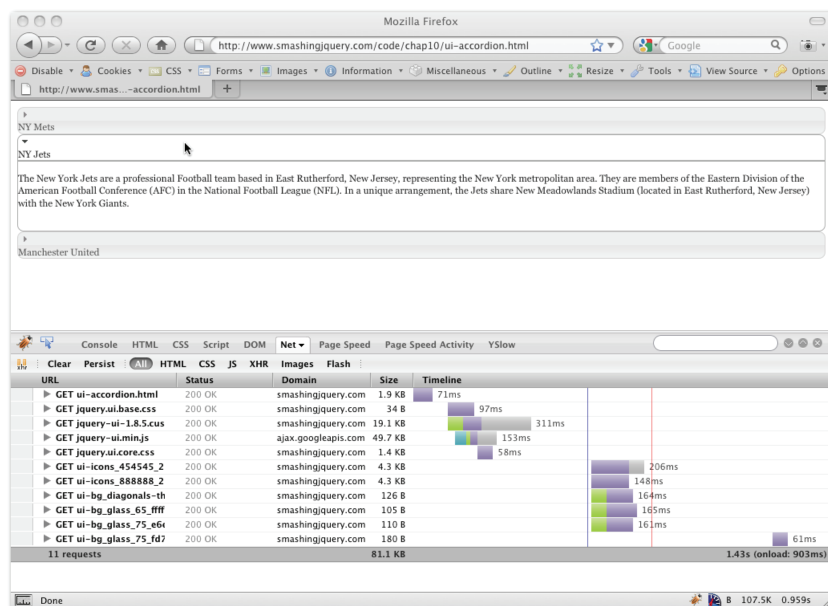


图10-11　jQuery UI 折叠组件示例（另见彩插图10-11）

如果你打算进一步定制Accordion，jQuery UI站点上有一份文档（http://jqueryui.com/demos/accordion/），这份文档含有accordion组件的所有可用选项。

3. Autocomplete组件

谷歌搜索引擎提供了自动完成功能（参见图10-12）。当我在谷歌搜索框中键入jquery UI时，谷歌会自动生成一个下拉菜单给出一些建议关键字供我选择。自动完成，又称为搜索提示，经常用于搜索输入框及一些其他类型的查找框。开始键入时，就开始进行逐字母匹配，而建议选项则直接显示在文本框之下，这极大地方便了用户的查找操作，特别是在用户对要查的单词拼写有点拿不准时。

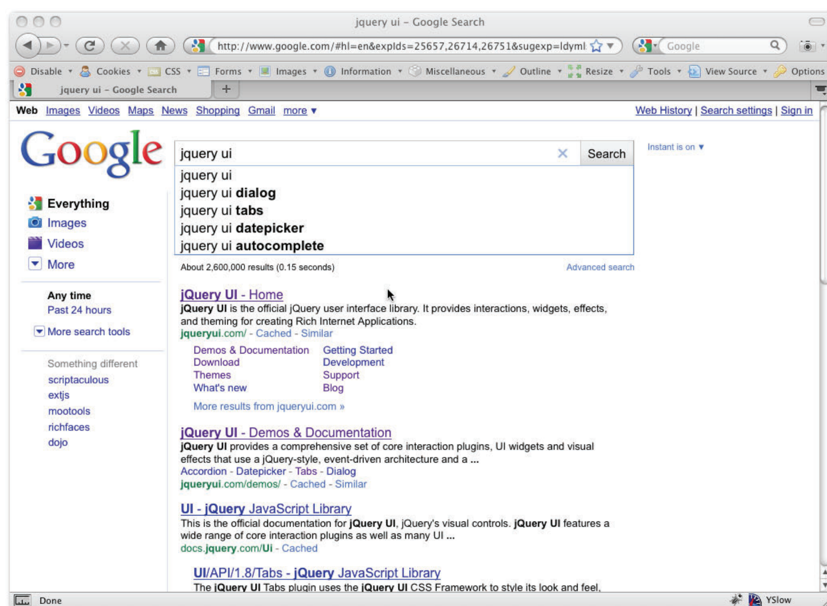


图10-12　谷歌搜索引擎提供的自动完成功能

jQuery UI中有一个用起来极其简单的自动完成组件，使用时只需要简单几行JavaScript代码。在下面这个例子中，我将介绍如何设置一个基础的自动完成组件（参见图10-13）。

(1) 通过Build Your Download页（http://jqueryui.com/download）下载包含Autocomplete组件的jQuery UI包，然后把jQuery UI包含到页首部分：

```
<script type="text/javascript" src="js/jquery-ui-1.8.4.custom.min.js"></script>
```

(2) 包含自定义主题的CSS文件：

```
<link type="text/css" href="../themes/base/jquery.ui.all.css" rel="stylesheet" />
```

(3) 创建一个`id="search"`的文本框，然后将Autocomplete组件绑定到这个文本框。

```
    <label>Search</label>
    <input type="text" id="search" />
```

(4) `.autocomplete()`方法有一个`source`选项，输入框内的文字就是和它进行匹配。你可以将这个`source`设置为远程脚本、一个XML文件或者像这个例子中一样设置为一个数组。要详细了解各种数据来源的类型，请参考jQuery UI的文档。

```
var availableTags = [
    "Nike",
    "Puma",
    "Aldo",
    "Tiger",
    "Cole Haan",
];
```

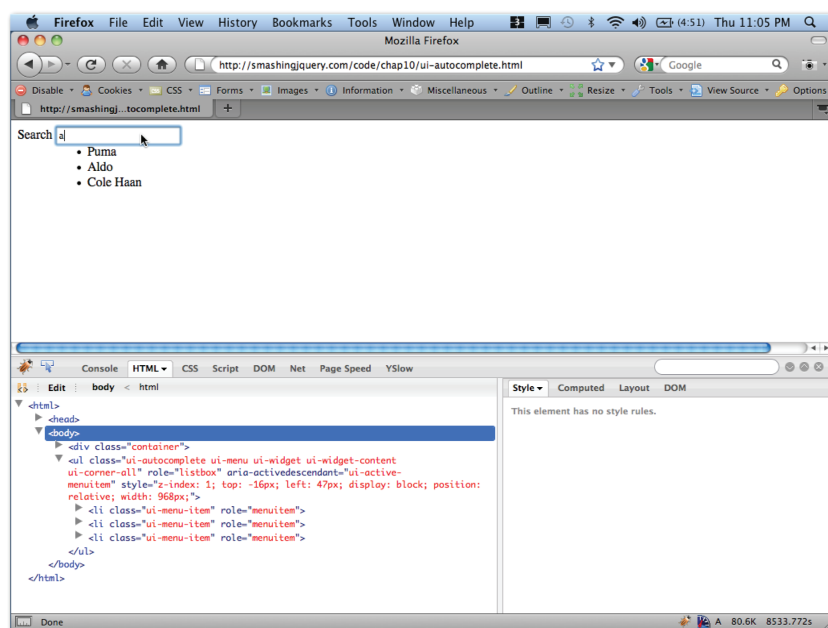


图10-13　一个jQuery UI 自动完成组件示例（另见彩插图10-13）

(5) 用jQuery选中`id="search"`的元素，然后调用`.autocomplete()`方法，并将上面创建的`availableTags`变量做为`source`参数传递进去。

```
$("#search").autocomplete({source: availableTags});
```

4. Datepicker组件

jQuery UI还包含一个相当灵活的日期拾取组件（创建一个小巧的弹出日历，供用户选取日期），可以绑定到任意一个文本输入框，就像图10-14中TripAdvisor网站上的日期拾取器一样。日期拾取器越来越流行，我甚至都记不起最后一次手工输入日期是什么时候了。

要是没有日期拾取器，网上的日子就不这么好过了。jQuery UI包含一个相当好用的日期拾取器组件。下面是一个日期拾取器教程（参见图10-15）。

(1) 通过Build Your Download页（http://jqueryui.com/download）下载包含Datepicker组件的jQuery UI包，然后把jQuery UI包含到页首部分：

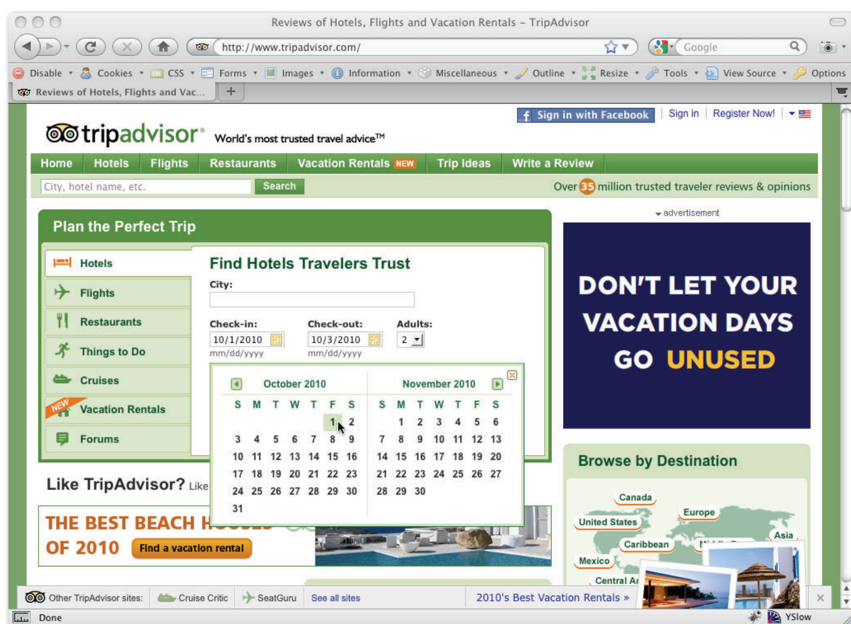
```
<script type="text/javascript" src="js/jquery-ui-1.8.4.custom.min.js"></script>
```

(2) 包含自定义主题的CSS文件：

```
<link type="text/css" href="../themes/base/jquery.ui.all.css" rel="stylesheet" />
```

(3) 创建一个`id="date"`的文本框，然后将它绑定到Datepicker组件。

```
<label>Date</label>
  <input type="text" id="date" />
  <div>
```



由TripAdvisor 公司许可复制

图10-14 TripAdvisor站点上应用的一个日期拾取器

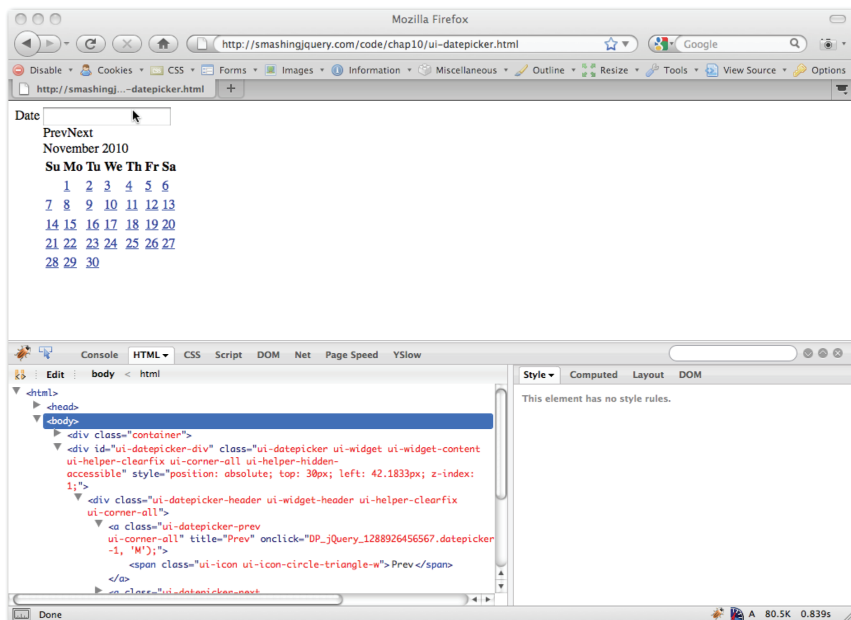


图10-15 jQuery UI日期拾取器

(4) 用jQuery选中`#date`元素，然后调用`.datepicker()`方法。要进一步定制Datepicker组件，请访问它的文档。文档地址为http://jqueryui.com/demos/datepicker/。

```
$('#datepicker').datepicker();
```

10.4　整合流行的 jQuery 插件到站点

无数的Web设计师和开发者都是jQuery的粉丝，他们在广阔的领域贡献了许多极其好用的插件(Ajax、图库等) 如果你打算写个什么东西，建议你先查一下，很可能会找到一个现成的jQuery插件（也很可能不止一个）。

jQuery 官 网 专 门 有 一 个 插 件 子 站 （ http://plugins.jquery.com/)，允许你提交自己编写的插件及针对自己站点所用第三方插件的错误报告（参见图10-16）。

插件能有效提高开发效率，减少开发时间。插件通常都有很好的文档，万一你遇到麻烦，并且无法联络作者以求得帮助，jQuery论坛(http://forum.jquery. com/) 是一个寻求支持和建议的好地方。

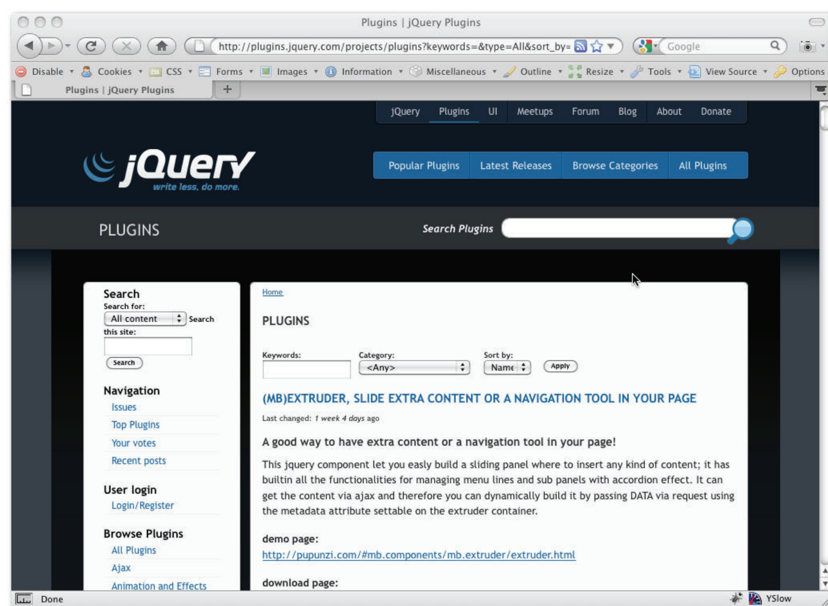


图10-16　jQuery官网插件子站

10.4.1　jQuery Tools

类似于jQuery UI， jQuery Tools（http://flowplayer.org/tools/index.html）也是一个UI库（参见图10-17），包含6个组件——tabs、tooltips、overlays、scrolling功能、forms和toolboxes。这个库体积非常小，只有大约16 KB，因此它是一个很好的jQueryUI替代品。jQuery UI拥有极其丰富的功

能，其中很多你可能一辈子也用不到。这意味着它的体积很大，会影响页面的加载速度及网站性能。如果你不需要使用所有的组件，这就有点讨厌。

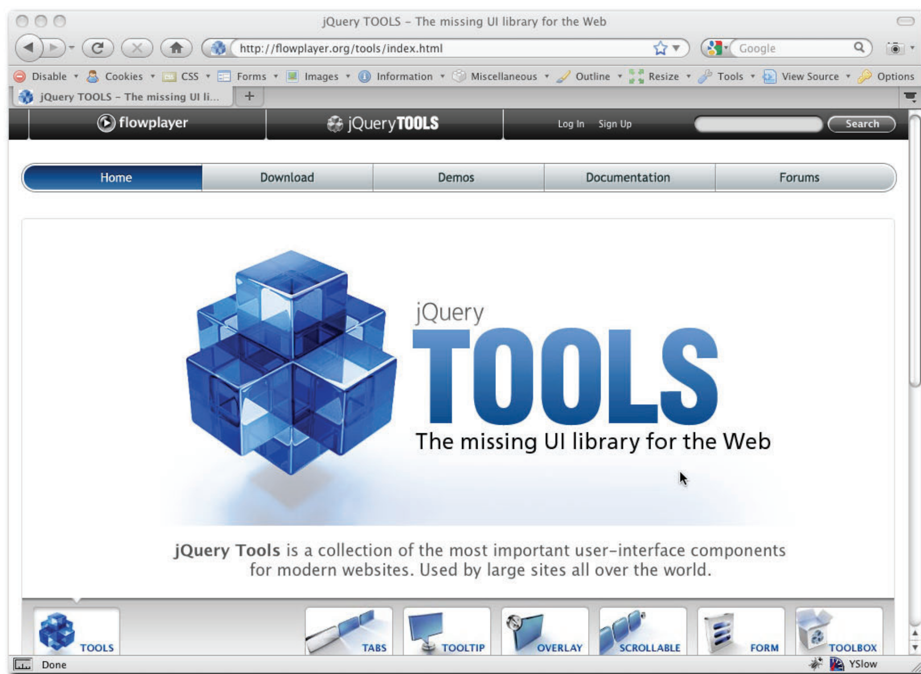


图10-17　jQuery Tools首页

jQuery Tools允许你只下载自己需要的组件，从而让库文件保持最小。

jQuery Tools是一个极其简化的UI库，因为它只包含6个组件，不提供主题库和CSS框架。

因此，修改jQuery Tools组件的外观不如使用jQuery UI时那样平滑。由于缺少CSS框架和主题库，你不得不根据站点或应用程序的需要亲自定义HTML和CSS。用jQuery Tools代替jQuery UI有很多好处，比如刚好够用的组件集（不必为不需要的功能买单）、更小的文件（约4KB），与已有HTML代码集成的能力（HTML代码不必遵守jQuery UI定义好的那一套结构）等具体因网站或应用而异。如果你是接手一个现成的站点，使用jQuery Tools（而不是jQuery UI）更容易改造或优化一些功能，比如标签和覆盖层。

1. 设置jQuery Tools

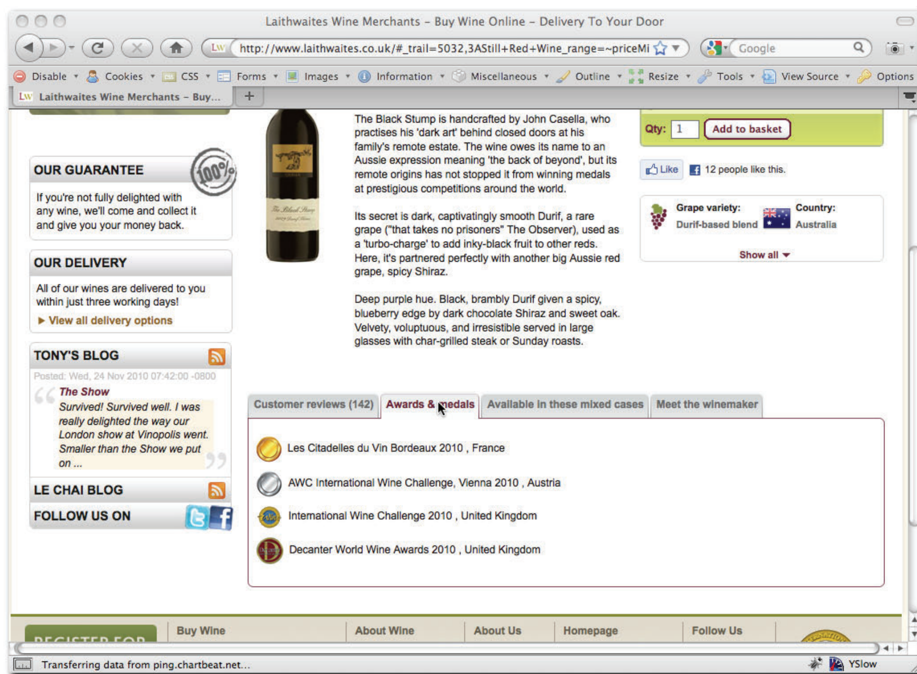
设置jQuery Tools的方式和设置jQuery UI几乎一样。你可以下载包含所有组件的整个包，也可以只下载自己需要的组件。

然后把下载完成的jQuery Tools Java Script文件上传到Web服务器的js目录下，并将如下所示的`script`标签放入你的网页，包含在jQuery底后面，其中路径为插件所在位置。

```
<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>
<script src="js/jquery.tools.min.js"></script>
```

2. 使用Tabs工具创建动态标签

我最近用Tabs工具在一个站点的产品详情页里添加了一个标签切换内容区，如图10-18所示。jQuery Tabs工具允许我们利用站点上已有的HTML结构快速添加标签。



由Laithwaites Wine许可复制

图10-18　使用了tabs组件的Laithwaites Wine网页

在下面这个教程里，我们使用jQuery Tools中的Tabs工具创建动态标签（类似前面jQuery UI的例子），只是这次我们对更少的HTML拥有更多的控制。这里提到更多的控制，说的是我们不必使用类似jQuery UI的CSS框架来设置标签的样式。

(1) 编写Tabs所需的HTML。在这个例子里，我使用一个无序列表保存标签标题，用一组嵌套的div元素保存标签内容。这些标签都能和jQuery Tools协同工作，因为后者不像jQuery UI那样严格要求特定的HTML结构。HTML结构由你定义。

```
<ul class="product-tabs">
  <li class="t-desc"><a href="#">Description</a></li>
  <li class="t-rev" ><a href="#">Customer Reviews</a></li>
</ul>

<div class="tab-panes">
```

```
    <div class="description">
      <p>Perfect with food, two presidents and the Queen of England.</p>
      <p>The "Rhône Rangers" are an elite group of California winemakers, centered
    around Paso Robles, whose holy grail is the dark, rich, black fruit magic of the best
    Rhône wines. Don Brady, elected 2006 Paso Robles Winemaker of the Year by his peers,
    has combined the keystone Rhône grapes of Syrah, Grenache and Cinsault to conjure
    his Cuvée de Robles 2006, exclusively for WSJwine. After 14 months in oak, with fresh
    cherry and cranberry aromas and rich, silky tannins, this is perfect wine to
    complement food and honor company. Which is why Don's wines were on the table at
    the White House dinner when the company was Queen Elizabeth II.</p>
    </div>

    <div class="customer-reviews" style="display:none;">
      <p>No Reviews Yet.</p>
    </div>

</div>
```

(2)用jQuery选中这个（.product-tabs）无序列表，然后应用.tabs()方法。我们必须把对应标签的选择器作为参数传递给.tabs()方法。在本例中，与标签相对应的内容封装在class="tab-panes"的元素中，因此我们应该使用".tab-panes > div"选择器匹配.tab-panes元素的所有div子元素。简单来说，这一语句的意思是：选中标签元素.product-tabs（标签导航），对其应用.tabs()方法并告诉.tabs()方法内容保存在.tab-panes中的div元素中。

```
$(".product-tabs").tabs(".tab-panes > div");
```

jQuery tools中的tabs组件有非常多的选项，可用来创建更高级的标签效果，比如带动画效果的幻灯片，支持淡入淡出效果、动态内容等。如果有兴趣详细了解这些选项，在jQuery Tools官方网站（http://flowplayer.org/tools/documentation/index.html）有相当完备的文档。

10.4.2　Fancybox

Fancybox（http://fancybox.net/）是一个基于jQuery库的轻量级lightbox插件。lightbox通过覆盖在当前页面上的覆盖层显示图片，这样就不必将用户重定向到新页面。当图片显示出来时，一个黑色的半透明背景覆盖在整个屏幕上，图片则通常位于一个白色边框的容器中并显示为前景图。图10-19展示了Netflix站点（www.netflix.com）上的一个lightbox效果。

Fancybox插件的故事始于最早的Lightbox插件。那还是在2006年，我刚刚开始使用Lightbox插件（www.lokeshdhakar.com/projects/lightbox2/）。从那时起，Lightbox陆续发布了很多新版本，其最近一次更新在2008年。由于Lightbox 2提供的技术支持有限，我转而开始使用Cody Lindley的Thickbox插件（http://jquery. com/demo/thickbox）。Thickbox尝试扩展了lightbox的功能，它不仅仅支持显示图片，而且支持显示视频，甚至来自其他地方的动态内容。在大约3年的时间里，我在很多项目中使用了Thickbox，直到Thickbox的作者停止了对它的支持。后来我选择了Fancybox（参见图10-20），它真是让我神清气爽。



图10-19　Netflix使用一个lightbox窗口显示产品简介（另见彩插图10-19）

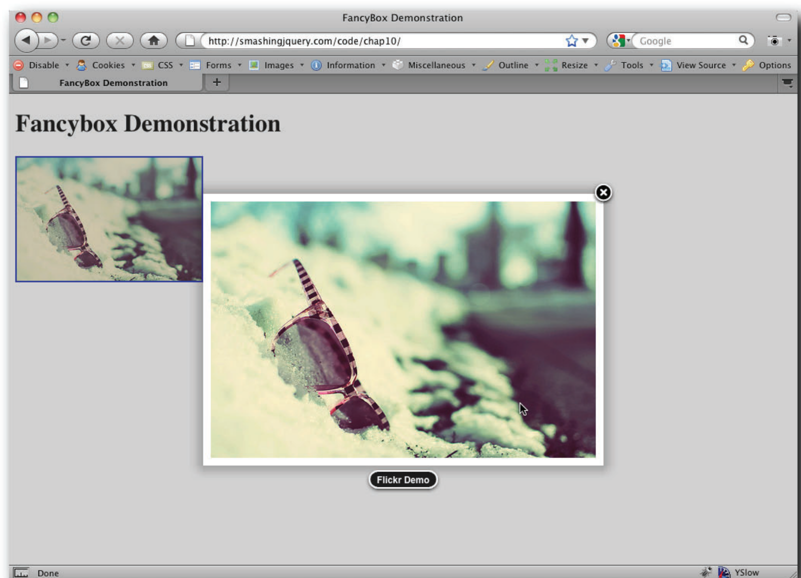


图10-20　Fancybox 插件演示（另见彩插图10-20）

我选择Lightbox插件的故事告诉我们：插件非常有用，只是一旦插件背后的某个人或某家公司对它停止了支持，你只能要么继续使用它，同时奢望自己在升级jQuery时该插件还能正常工作，要么扔掉该插件再找一个新的。每当这一幕上演时，我都选择一个新插件，通常是一个更好，并且具有更多功能的新插件。

(1) 在使用jQuery插件时，总要将它包含在页首，且总是将插件包含在紧跟jQuery库之后（务必将jQuery库和插件包含在自己的jQuery代码之前）。

```
<script src="js/fancybox.js"></script>
```

(2) 准备HTML代码时，为你准备应用Fancybox插件的链接标签添加一个类。类的名字可以随便取，我在此处为其取名为`fancybox`。

```
<a class="fancybox"
href="http://farm5.static.flickr.com/4058/4252054277_f0fa91e026.jpg"><img alt=
"example1" src="http://farm5.static.flickr.com/4058/4252054277_f0fa91e026_m.jpg" />
</a>
```

(3) 写一个语句选中`.fancybox`元素，然后调用`.fancybox()`方法。如果你不提供任何参数，`.fancybox()`方法就使用默认设置。

```
$(".fancybox").fancybox();
```

请访问Fancybox官方站点（http://fancybox.net/）查看完备的文档。

10.5　编写第一个 jQuery 插件

要把自己编写的功能代码分享到jQuery社区，最好的方式是写一个插件，这样开发者们就能非常容易地将它应用到自己的项目中。如果一个站点中好多应用都要用到某个功能，你也可以把这个功能写成插件自己用。

在构建与维护DOM或Ajax迭代过程中，绝大多数网站问题或功能改进都可用jQuery解决，并可随后将解决问题的代码改写成一个插件，以便重用。不过，当已经有类似功能的插件可用时，或者你的代码仅仅是一次性用来修复一个小bug，就没有必要为写插件而写插件了。在编写jQuery插件时，请务必牢记下面几个重要的提示。

- 为插件取一个独一无二的名字。在选定名字之前，一定要检查一下是否和已有插件重名。
- 为了避免和其他也使用了`$`别名的JavaScript库发生命名冲突，在插件中要避免使用`$`别名。
- 合理注释你的插件。
- 一定要为插件选一个开源许可协议，以方便插件传播。
- 动手写代码之前，先在纸上做计划，把插件要干什么以及打算怎么干想清楚。
- 要有默认选项，且默认选项可被用户设置覆盖。
- 阅读jQuery插件文档，检查自己的插件是否符合要求，参见图10-21（http://docs.jquery.com/Plug-ins/Authoring）。

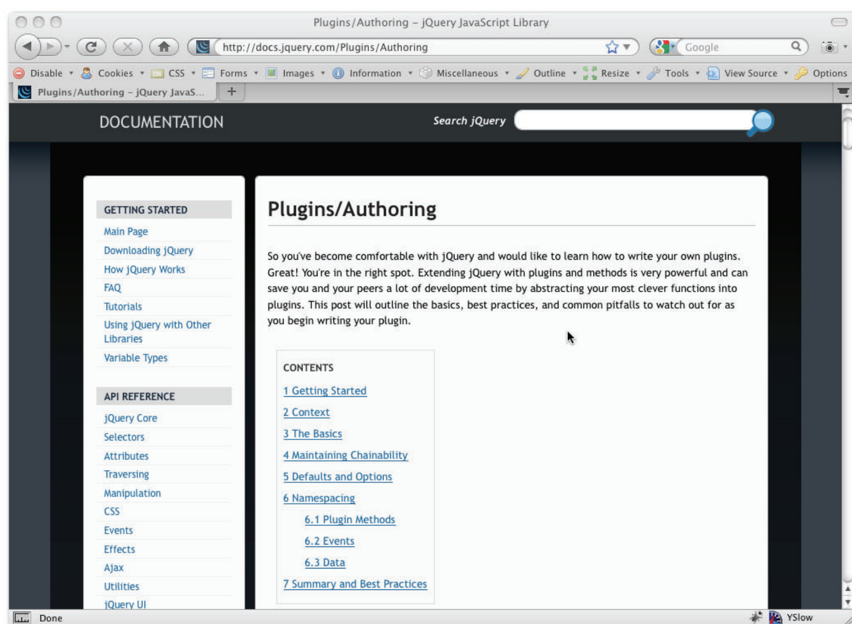


图10-21　jQuery插件文档

10.5.1　筹划一个插件

接下来我要写一个示例插件，在这个例子中，我打算选用一个动态生成的无序列表，让这个列表的前10项可见，剩下的项处于隐藏状态，使用一个简单的显示/隐藏导航控制隐藏项是否显示。我希望它能改变导航链接的文本，也能改变显示出来的链接数目。这些是相当基础的功能，非常适合用来展示如何编写插件。嗯，它很简单，不至于把大家搞糊涂。

10.5.2　插件的结构

在开始编写插件之前，我要介绍一下插件的基本结构。所有的jQuery插件都声明为`jQuery.fn`对象的方法：

```
jQuery.fn.showhidePlugin = function() {
 //插件相关代码
};
```

我们使用如下代码使用这个插件：

```
$('.selector').showhidePlugin();
```

为了和已有插件保持一致，我建议在保存插件时采用jquery.pluginname.js这种命名格式。我总是在插件文件名的开头使用单词jquery，这非常有利于代码文件的组织，让人一眼就能看出它

是一个jQuery插件。

注意我没有使用$别名，而是使用jQuery以避免任何可能的命名冲突。如果更喜欢使用$而不是jQuery，可以像下面这样把插件代码封装在一个自执行的匿名函数中。这个方法非常巧妙，仅仅增加了一个自执行函数，就能够安全地在插件内使用$别名，不会引起任何问题。

```
(function($){
jQuery.fn.showhidePlugin = function() {
 //插件相关代码
};
})( jQuery );
```

10.5.3　设定插件选项

通过传递给插件方法一个options参数，我们就可以让插件支持设置选项。我们使用一个JSON对象作为插件的默认选项。这个插件有3个默认选项——numShown、showText和hideText。默认设置在插件内部初始化，运行时传递进来的用户选项会覆盖掉默认设置。

```
(function($){
jQuery.fn.showhidePlugin = function(options) {

//设定默认值，用逗号分隔选项
var defaults = {
  numShown: 10,
  showText : 'Show More Links',
  hideText : 'Hide Links'
}

};
})( jQuery );
```

接下来，使用$.extend方法合并默认选项和用户选项。如果调用时提供了用户选项，它们将覆盖掉插件内部声明的默认选项。

> .extend()允许你使用一个或多个对象扩展基准对象，扩展的方式是依序将右边的对象合并到基准对象（左边第一个对象）。

```
(function($){
jQuery.fn.showhidePlugin = function(options)
{

//设定默认值，用逗号分隔选项
var defaults = {
    numShown: 10,
    showText : 'Show More Links',
    hideText : 'Hide Links'
}

var options =  $.extend(defaults, options);

};
})( jQuery );
```

10.5.4 创建插件

我们在上一步中完成了选项设置，接下来将混合使用jQuery脚本和JavaScript原生脚本实现插件的主体功能。这个插件将适用于任何无序列表。我们使用numShown变量的值截断这个列表，然后添加隐藏和显示链接来切换隐藏项的显示状态（可见或隐藏）。

(1) 添加一个.each()方法迭代选择器匹配的元素，并返回this对象。这一步（返回this对象）非常重要，所有的插件都要有这一步才能正常工作。在一个无序列表上应用该插件时，.each()方法就会遍历那个无序列表的所有项。

```
return this.each(function() {
});
```

(2) 创建两个变量——o和obj。o引用options，obj引用$(this)。有了这两个变量，后面引用起它们来就省很多事。变量obj中保存的是包裹着当前无序列表元素的jQuery对象。

```
return this.each(function() {
  var o = options;
  var obj = $(this);
});
```

(3) 接下来我们创建3个变量——pLength、numHidden和pList。pLength是obj包含全部子元素的总数，numHidden由pLength减去o.numShown（这是一个设置选项）得来。如果pLength是30，而o.numShown等于5，则numHidden的值就是25。变量pList保存着obj对象的所有子元素。

```
return this.each(function() {
  var o = options;
  var obj = $(this);

  //确定项数并计算被隐藏项的数目
  var numHidden = pLength - o.numShown;
  var pList = obj.children();

});
```

(4) 再创建一个变量shLink，这个变量保存着一段HTML代码（一个链接），这段代码用来显示控制目标文本显示或隐藏的链接文本。这个链接的文本来自默认选项中的showText选项，并且具有一个类.view，我们稍后会详细解释这个类。我们用这个链接控制目标文本的显示/隐藏。

```
return this.each(function() {
  var o = options;
  var obj = $(this);

 //确定项数并计算被隐藏项的数目
  var numHidden = pLength - o.numShown;
  var pList = obj.children();

 //设置显示/隐藏链接
  var shLink = "<a href='#' class='view'>" + o.showText + "</a>";
  });
```

(5) 添加一个条件语句判断pLength是否大于o.numShown，如果大于则在obj对象前插入shLink。这在需要时就会把显示/隐藏链接放到当前无序列表之前。

```
return this.each(function() {
  var o = options;
  var obj = $(this);

 //确定项数并计算被隐藏项的数目
  var numHidden = pLength - o.numShown;
  var pList = obj.children();

 //设置显示/隐藏链接
  var shLink = "<a href='#' class='view'>" + o.showText + "</a>";

  if (pLength > o.numShown) {
    jQuery(shLink).insertBefore(obj);
  }
});
```

(6) 然后对pList应用一个.each()循环，迭代处理当前无序列表的所有子元素。

```
return this.each(function() {
  var o = options;
  var obj = $(this);

 //确定项数并计算被隐藏项的数目
  var numHidden = pLength - o.numShown;
  var pList = obj.children();

 //设置显示/隐藏链接
  var shLink = "<a href='#' class='view'>" + o.showText + "</a>";

  if (pLength > o.numShown) {
  jQuery(shLink).insertBefore(obj);
  }
  pList.each(function(index){
  });
});
```

(7) 在刚刚创建的.each()循环的回调函数中，添加一个if/else条件语句，检查index是否小于o.numShown。如果小于，则显示当前迭代元素，否则添一个hidden类，将其隐藏。index变量是该循环的迭代变量，每循环一次它（从0开始）都自动加1。

```
return this.each(function() {
  var o = options;
  var obj = $(this);

  //确定项数并计算被隐藏项的数目
  var numHidden = pLength - o.numShown;
  var pList = obj.children();

  //设置显示/隐藏链接
  var shLink = "<a href='#' class='view'>" + o.showText + "</a>";

  if (pLength > o.numShown) {
  jQuery(shLink).insertBefore(obj);
```

10

```
        }
        pList.each(function(index){
          if (index < o.numShown) {
            jQuery(pList[index]).show();
        }
        else {
          jQuery(pList[index]).hide();
          jQuery(pList[index]).addClass('hidden');
        }
        });
      });
```

(8) 使用.live()方法为a.view绑定click事件处理函数。在处理函数末尾使用return false避免触发click事件的默认行为（重定向）。同时加一个语句选中所有带有.hidden类的元素，切换其显示状态（隐藏/显示）。

```
return this.each(function() {
  var o = options;
  var obj = $(this);

 //确定项数并计算被隐藏项的数目
  var numHidden = pLength - o.numShown;
  var pList = obj.children();

 //设置显示/隐藏链接
  var shLink = "<a href='#' class='view'>" + o.showText + "</a>";

  if (pLength > o.numShown) {
    jQuery(shLink).insertBefore(obj);
  }
  pList.each(function(index){
    if (index < o.numShown) {
    //alert('test');
    jQuery(pList[index]).show();
    } else {
      jQuery(pList[index]).hide();
      jQuery(pList[index]).addClass('hidden');
     }
  });
  //切换文本
  jQuery("a.view").live("click", function(e){
    jQuery('.hidden').toggle();
    return false;
  });
});
```

(9) 最后一步，添加一个条件语句切换显示/隐藏链接的链接文本。用if语句检查a.view链接文本是否与o.showText相同，如果相同，则在单击它时将它的链接文本变为o.hideText，如果不同，则在单击它时将它的链接文本变为o.showText。

```
return this.each(function() {
  var o = options;
  var obj = $(this);

  //确定项数并计算被隐藏项的数目
```

```
  var numHidden = pLength - o.numShown;
  var pList = obj.children();

  //设置显示/隐藏链接
  var shLink = "<a href='#' class='view'>" + o.showText + "</a>";

  if (pLength > o.numShown) {
    jQuery(shLink).insertBefore(obj);
  }
  pList.each(function(index){
    if (index < o.numShown) {
    //alert('test');
      jQuery(pList[index]).show();
  } else {
      jQuery(pList[index]).hide();
      jQuery(pList[index]).addClass('hidden');
    }
    });
  //切换文本
  jQuery("a.view").live("click", function(e){
    if (jQuery(this).text()==o.showText) {
       jQuery(this).text(o.hideText);
} else {
       jQuery(this).text(o.showText);
    }
   jQuery('.hidden').toggle();
   return false;
  });
});
```

下面列出了插件 jquery.showhideplugin.js的全部源代码：

```
(function($){
jQuery.fn.showhidePlugin= function(options) {

//设置默认值，用逗号分隔选项
var defaults = {
    numShown: 10,
    showText : 'Show More Links',
    hideText : 'Hide Links'
}

var options =  $.extend(defaults, options);

return this.each(function() {
var o = options;
var obj = $(this);

//确定项数并计算被隐藏项的数目
var pLength = obj.children().length;
var numHidden = pLength - o.numShown;
var pList = obj.children();

//设置显示/隐藏链接
```

10

```
var shLink = "<a href='#' class='view'>" + o.showText + "</a>";

if (pLength > o.numShown) {
  jQuery(shLink).insertBefore(obj);
}

pList.each(function(index){
  if (index < o.numShown) {
    jQuery(pList[index]).show();
} else {
    jQuery(pList[index]).hide();
    jQuery(pList[index]).addClass('hidden');
  }
});

//切换文本
jQuery("a.view").live("click", function(e){

  if (jQuery(this).text()==o.showText) {
     jQuery(this).text(o.hideText);
} else {
     jQuery(this).text(o.showText);
}
   jQuery('.hidden').toggle();
   return false;
  });

});
}
})(jQuery);
```

插件现在完成了，接下来将它应用到我们的站点上：找一个元素，应该该插件，看看它是否工作。为此我创建了一个简单的无序列表，它包含十几个列表项。

(1) 准备测试插件用无序列表的HTML。在这个例子里，我使用了一个带有`.big-list`类的无序列表。

```
<ul class="big-list">
   <li>Test 1.</li>
   <li>Test 2.</li>
   <li>Test 3.</li>
   <li>Test 4.</li>
   <li>Test 5.</li>
   <li>Test 6.</li>
   <li>Test 7.</li>
   <li>Test 8.</li>
   <li>Test 9.</li>
   <li>Test 432.</li>
   <li>Test 23.</li>
   <li>Test 0232.</li>
   <li>Test 2002.</li>
   <li>Test 541.</li>
   <li>Test 5432.</li>
   <li>Test 542.</li>
```

```
        <li>Test 542.</li>
        <li>Test 342.</li>
        <li>Test 452.</li>
        <li>Test 42.</li>
        <li>Test 542.</li>
        <li>Test 4542.</li>
        <li>Test 432.</li>
        <li>Test 23.</li>
        <li>Test 0232.</li>
        <li>Test 2002.</li>
</ul>
```

(2) 在页面的头部，jQuery库之后包含插件文件：

```
<script src="js/jquery.showhidePlugin.js" type="text/javascript"></script>
```

(3) 用jQuery选中.big-list元素，然后调用showhidePlugin()方法。因为我们没有为showhidePlugin()方法提供任何选项，所以showhidePlugin()会使用默认设置。

```
$(".big-list").showhidePlugin();
```

(4) 如果你想改变如图10-22所示显示/隐藏插件的显示，只需要用一个对象字面量（而非每个参数）将需要改变的所有默认参数传递给showhidePlugin()方法。

```
$(".big-list").showhidePlugin({
  numShown:15,
  showText:"Show FTW"
});
```

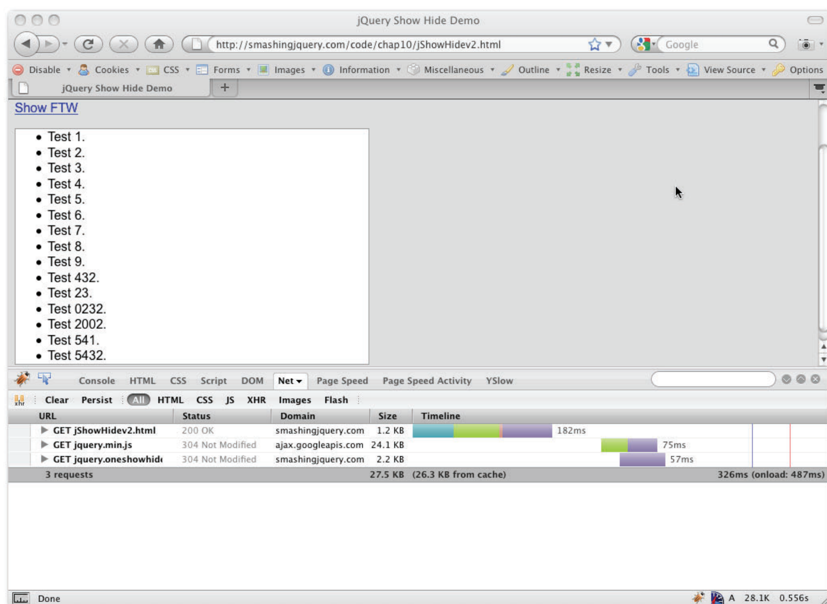


图10-22　自定义jQuery 显示/隐藏插件及其选项的应用示例

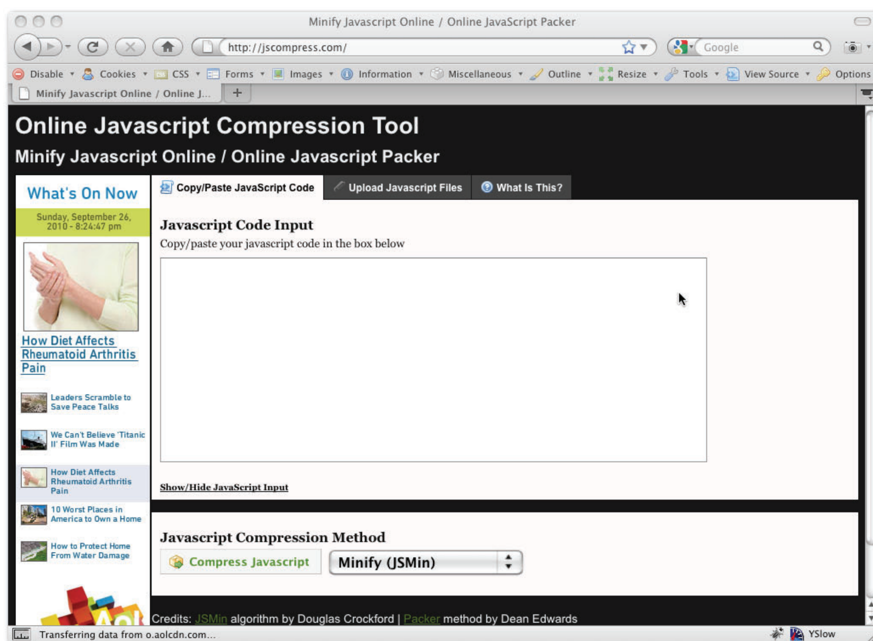
10.6　如何发布 jQuery 插件

准备发布插件时，有好几个地方需要注意。在正式放出你的插件之前，请务必挨个检查以下3点：

- 测试插件是否在所有浏览器下都能正常工作，这里说的浏览器包括IE6、IE7、IE8、Firefox、Safari、Chrome还有Opera；
- 写一份文档解释选项的含义以及插件的工作原理；
- 如果你愿意开放该插件的源代码，就到开放源代码官方站点（www.opensource.org/licenses/index.html）为插件选择一份开源许可协议。

10.6.1　打包插件以便发布

打包前请一定确认插件代码可用于生产环境。你可以使用Douglas Crockford的代码压缩器、JavaScript Packer（http://jscompress.com/）等代码压缩工具最化小你的代码（参见图10-23）。最小化技术可删除代码中不必要的字符，比如注释、换行符、多余的空格和制表符，因此能（通过减小代码体积）加快代码加载。同时提供开发版（未压缩代码）和压缩版代码总是一个好主意。如果你的插件非常小（未压缩时小于4 KB），压缩就不是必需的，不过在发布前压缩代码对我已是习惯。



Douglas Crockford负责JSMin算法 | Dean Edwards负责压缩方法 | Vance Lucas负责JavaScript压缩工具/服务

图10-23　Douglas Crockford的最小化代码脚本

10.6.2　发布插件

你可以将自己的插件发布到一些站点，一方面可知道这插件是否很成功，另一方面也是对开源社区的回报。如果你的插件是开源的，我强烈建议把它发布在以下地方。

- **jQuery插件子站**　jQuery官方站点提供了一个wiki（http://plugins.jquery.com/），在那里开发者可以提交自己的代码。你可以通过浏览器上传插件，也可以通过版本控制工具（比如Subversion或Git）上传。在那儿你可以为上传的插件指定一个名字及相应的描述，也可以回复用户提交的bug报告，还可以查看社区对该插件的评级，发表插件文档等。
- **谷歌的项目托管站点**　Google Code（http://code.google.com/hosting/）提供的服务类似于jQuery 插件子站。除了便利的服务之外，把插件托管到Google Code最大的好处是：因为服务提供商是谷歌，所以几乎没有流量限制。

第 11 章 jQuery在移动Web开发中的应用

我们可以在任何网站或应用中使用jQuery，移动Web应用也不例外。手机有屏幕大小和带宽限制，这些限制在移动Web开发过程中要始终牢记在心。移动Web应用既可以是通过浏览器或移动设备访问的纯Web应用，也可以是原生的Web应用程序。

在本章中，我将解释移动Web站点与原生移动应用二者的不同，讨论CSS3和HTML5为移动Web开发带来的益处，并告诉大家如何安装和设置（适用于Google Android及苹果移动设备的）移动浏览器模拟器。最后，我将和大家一起看看jQuery Mobile Alpha为移动Web开发带来了什么好处。

11.1 使用 jQuery 构建移动 Web 应用

开发移动Web应用有好几条途径可选。移动Web应用市场似乎高度集中于两大平台——Apple iPhone/iPad 和 Google Android。

我们在开发兼容多种移动浏览器的Web站点或应用的时候，不难发现移动Web开发比桌面（基于浏览器的）开发困难得多。对jQuery开发者来说，移动Web开发还是一个相当新的领域。这主要是因为市面上流行的手机中，大部分对JavaScript的支持很差。不过苹果的iPhone和谷歌的Android智能手机的浏览器拥有强大的功能，为这些平台开发应用最容易。

下面是移动开发的两个小窍门。

- 开发之前定义目标用户。也就是事先确定该应用的目标人群。这些人使用哪些平台？他们在这些平台上使用什么浏览器？他们使用Wi-Fi还是蜂窝式网络？
- 让站点或应用保持简单。注意，移动设备带宽有限，屏幕通常也比较小。

图11-1展示的是Roanoke 学院站点的移动版，它是一个为iPhone浏览器设计站点的好例子，使用了jQuery框架，用其Ajax方法`$.load()`载入内容。它还使用了一个jQuery插件，以类似幻灯片的方式展示该学校近期的一些图片。



图11-1　Roanoke学院站点移动版，使用了jQuery框架

11.2　移动浏览器

多年前，桌面上发生了一场激烈的浏览器大战，战火绵延至今，这场战争发生在Internet Explorer、Mozilla Firefox、Apple Safari、Opera，还有新来者Google Chrome之间。一场类似的战争正在移动浏览器中上演，不同的移动浏览器有着不同的身手。不过随着移动Web应用市场的繁荣，不同的浏览器将会支持更多相同的功能。

不必写一行原生移动平台代码，我们就能使用熟悉的HTML、CSS和jQuery工具开发移动浏览器应用。苹果公司的iPhone iOS和谷歌的Android这两大智能手机平台，分别内建了相当先进的移动浏览器。这些设备对HTML5、CSS3和JavaScript的支持情况见表11-1。原生应用程序是为手机操作系统开发的软件。相对于Web站点或Web应用来说，原生应用确实有极大的优势，不过我们在浏览器平台上一样能开发出有价值的东西。如果你有开发桌面Web应用的经验，那就更没有问题。

表11-1　移动浏览器的功能

	HTML5	CSS3	JavaScript
Apple Safari 5	是	是	是
Google Android（Chrome）	是	是	是
Mozilla Firefox Mobile	是	是	是
Opera 10 Mobile	否	是	是

如表11-1所示，Safari、Chrome、Firefox Mobile对最新技术均有着良好的支持，其中Safari和Chrome要优于Firefox Mobile，因为它们分别是iPhone和Android的默认浏览器。在谷歌Android和苹果iPhone平台上也可以下载到Opera 10 Mobile浏览器。

这两大平台都支持CSS3和HTML5。如果你还没听说过这些高级Web设计技术，这次就能大饱眼福了！由于它们关系相当密切，人们在提到CSS3的时候，几乎一定也会提到HTML5。不过，这两项技术之间有着许多不同。

11.2.1　CSS3

CSS3（也就是层叠样式表3）早在2005年12月就开始开发了。现在，越来越多先进的浏览器（比如Apple Safari 4、Google Chrome、Microsoft IE 9、Opera 10，还有Firefox 4）开始支持CSS3。随着这些浏览器占领越来越大的市场份额，CSS3也逐渐成为主流。现在，我们能够利用这些技术在iPhone和Android平台上构建移动Web站点了。使用CSS3这种最新、最优的技术开发移动Web站点和应用的经验不仅能提高你的技术水平，为你的简历添上精彩一笔，还能增进你对移动开发限制的理解。

下面列出了CSS3提供的一些新属性，要查看更完整的新属性列表，可访问www.css3.info（参见图11-2）。这些属性特别适用于移动设备，要知道在过去需要综合应用图片、CSS变通方案和JavaScript才能实现这些效果，并且这些技术会因为增加页面的体积造成性能低下。使用这些CSS属性，能有效地保障移动Web站点和应用的性能。

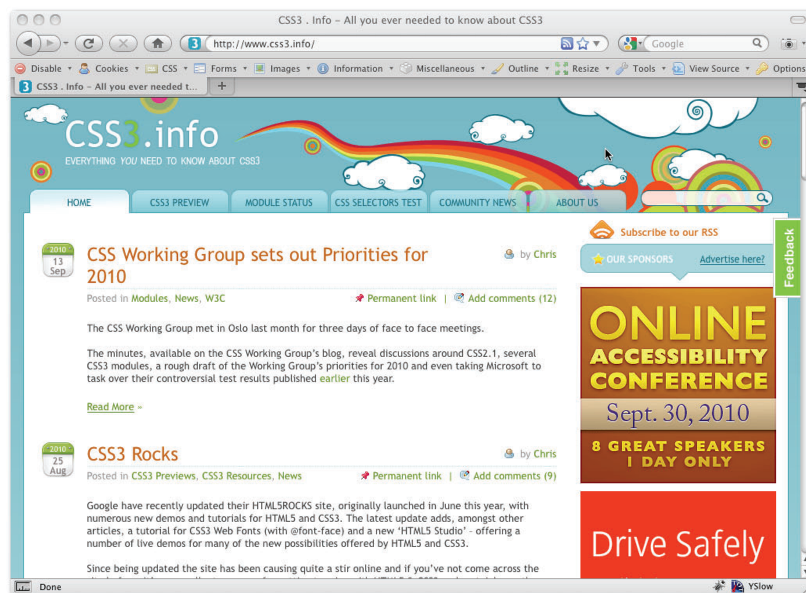


图11-2　CSS3资源站CSS3.info

- ❑ Border Radius　使用CSS创建圆角效果（参见图11-3）。
- ❑ Border Image　使用图片创建边框。
- ❑ Border Shadow　使用CSS生成阴影效果。
- ❑ Multiple Backgrounds　使用更少的代码支持多重背景，并用CSS声明一个多重背景定位属性。
- ❑ New Color Options　（不同于RGB和16进制的）新的颜色选项，提供真透明支持。
- ❑ Text Shadow　为文本元素创建阴影效果。

如果你想了解CSS3的所有新功能及特性，请访问www.css3.info/preview/。顺便说一下，jQuery支持CSS3选择器。

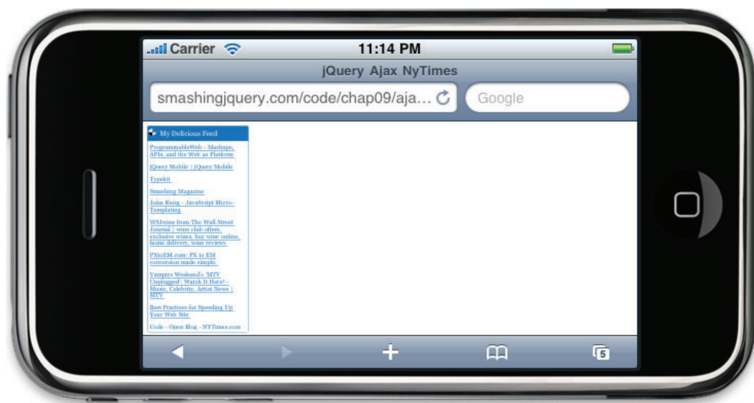


图11-3　iPhone模拟器中CSS3 Border Radius属性的效果

11.2.2　HTML5

HTML5的开发始于2008年1月，这些年来逐步得到了主流浏览器的支持。下面列出了HTML5提供的部分特性：

- ❑ 内建表单验证支持；
- ❑ 本地存储；
- ❑ header和footer标签，使内容得以更好地组织，而不必再依赖div标签；
- ❑ audio/video标签；
- ❑ canvas标签，动态创建图片或图画；
- ❑ 地理定位API（Geolocation API）。

> 这些功能为Web开发带来了极大的便利，不过HTML5提供的功能远比我提到的这些多得多。随着浏览器对HTML5和CSS3的支持日渐成熟，未来的Web设计必将越来越迷人。

11.2.3　移动开发的必要装备

移动开发的难点在于决定支持哪些平台以及如何测试。我着重支持Google Android和Apple

iPhone。为了方便（测试），你可以购买一部Android手机和一部iPhone，但这要花费大量的金钱，还要投入很多的时间。其实大可不必这样做，苹果和谷歌这两家公司都提供了开发者门户网站和包含模拟器的开发工具包（SDK）。所谓模拟器就是一个桌面程序，它模拟一台真实的Android或iPhone设备，既可用来测试原生应用，也可测试Web应用和网站。

1. 下载Apple iPhone Safari桌面模拟器

在developer.apple.com/devcenter/ios/index.action 可下载苹果的Safari模拟器。需要先在Apple.com开发者门户注册成为开发者，你才能下载并安装iPhone模拟器。模拟器安装好之后，你就可以用它通过Safari浏览任何网站，就像手里拿着一个真正的iPhone一样（参见图11-4）。



图11-4　苹果iPhone Safari 桌面模拟器

2. 下载Google Android桌面模拟器

可以从developer.android.com/sdk/index.html页面下载Google Android模拟器。下载并安装好Android Chrome模拟器之后，你就可以用它通过Chrome Mobile浏览任何网站，就像手里拿着一个真正的Android手机一样（参见图11-5）。

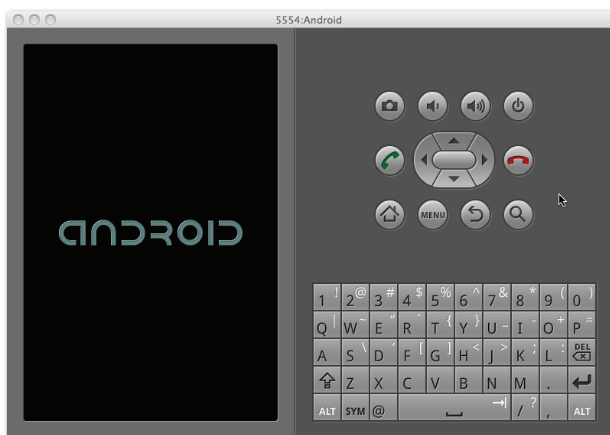


图11-5　Google Android桌面模拟器

11.2.4　面向Apple iPhone Safari移动浏览器的开发

为iPhone Safari浏览器开发Web站点或应用真的非常爽，这得益于Safari浏览器棒极了的性能（参见图11-6）。Safari支持CSS3和HTML5，因此你该知道刚才我为什么那么激动了。非要说它不好的地方，那就是只有包含Safari Mobile在内的少数几款浏览器支持CSS3和HTML5，因此有时候不得不在支持哪些特性和移动设备方面花点心思。

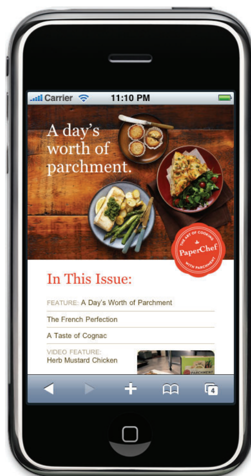


图11-6　苹果公司iPhone手机上的Safari游览器和Web应用

2007年6月，当苹果公司的iPhone首次发布时，iPhone还没有一个原生应用程序。那时候Web应用程序独领风骚，苹果公司为此还专门在iPhone站点上开辟了一个区域，展示那些精心为iPhone开发的Web应用（参见图11-7）。

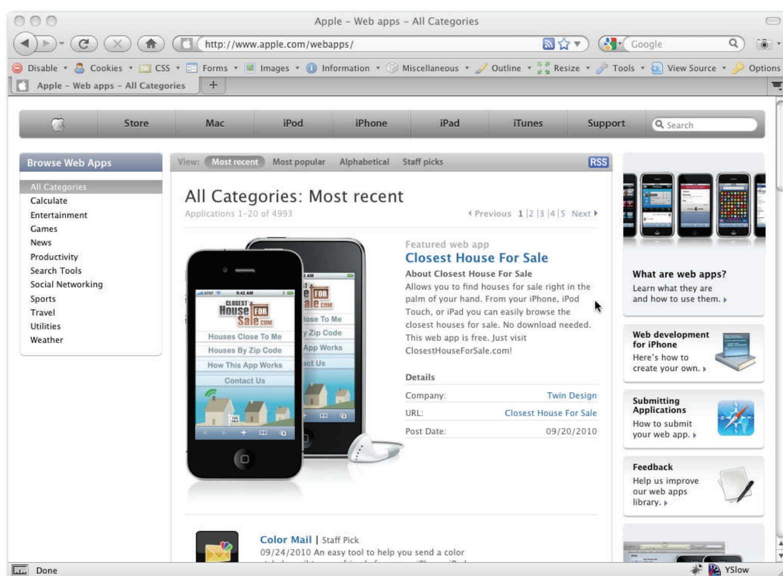


图11-7　Apple iPhone Web应用

为促进原生应用和Safari Web应用的发展，苹果公司在创建开发者门户网站上下足了工夫。在进行开发之前，我极力推荐你访问Safari开发中心（Safari Dev Center，参见图11-8）。开发中心提供了视频、技术文档以及示例代码，帮助我们学习为iPhone开发Safari Web应用程序。

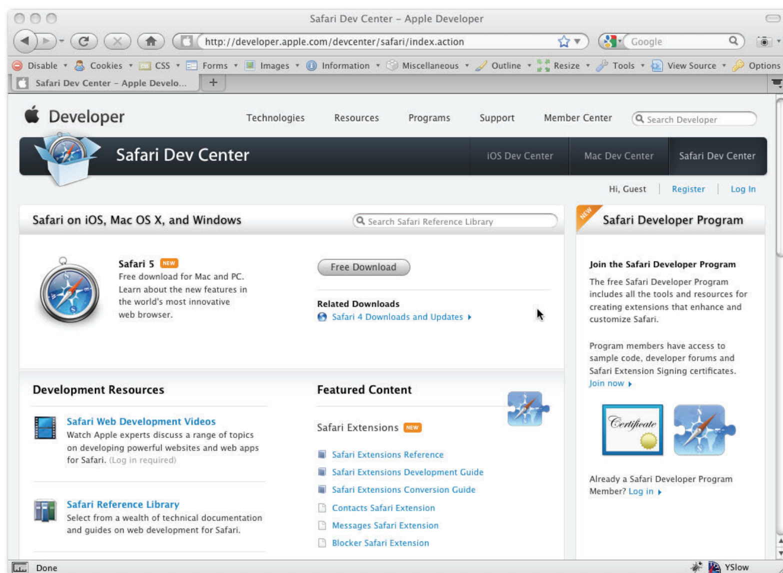


图11-8　Safari Dev Center

11.2.5　面向Google Android的Chrome浏览器的开发

为Android上的Google Chrome浏览器（参见图11-9）开发移动应用一样激动人心，因为它也支持CSS3和HTML5。Android也支持jQuery，因此桌面Web站点或应用的代码能非常容易移植到移动项目。

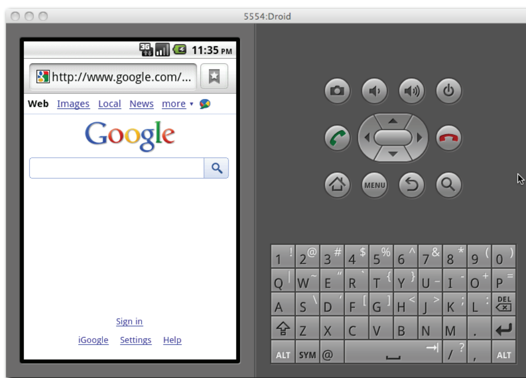


图11-9　Google Android Chrome浏览器

谷歌为打造一个卓越的开发者门户网站费尽心血，类似于iOS开发中心，谷歌也提供了视频、技术文档、代码示例以及论坛。Google Android开发中心（参见图11-10，http:// developer.android.com/index.html）提供了相当丰富的信息，在开始任何Web应用开发工作之前，绝对应该先浏览一下这些信息。

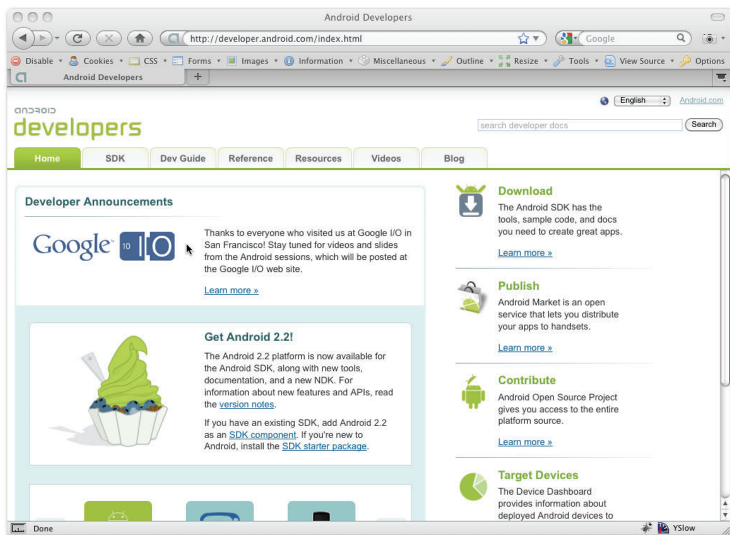


图11-10　Google Android开发中心

11.2.6　在不同智能手机上显示不同内容

使用Web服务器的URL重写功能，我们可以探测用户使用的是桌面浏览器还是移动浏览器，然后把用户重定向到专门为该平台开发的站点。下面的代码演示了如何为不同的移动平台进行重定向。每一行都使用了一个重写规则来检查用户代理，看看它是iPhone、黑莓还是Android。一旦匹配成功，用户就会被重定向到适合他手机的移动站点。这些代码使用了`Mod_rewrite`模块支持的正则表达式语法，为不同的移动平台设置不同的规则。

在使用以下代码之前，你需要确认使用的Web服务器是否是Apache且已启用`Mod_rewrite`模块。如果站点使用的是Windows服务器，尽管可以通过谷歌搜索服务找到替代方案，但最好还是问问你的Web主机提供商，他们能够为你指出正确方向。

```
RewriteCond %{HTTP_USER_AGENT} ^.*iPhone.*$
RewriteRule ^(.*)$ http://iphone.yourdomain.com [R=301]
RewriteCond %{HTTP_USER_AGENT} ^.*BlackBerry.*$
RewriteRule ^(.*)$ http://bb.yourdomain.com [R=301]
RewriteCond %{HTTP_USER_AGENT} ^.*Android.*$
RewriteRule ^(.*)$ http://android.yourdomain.com [R=301]
```

11.2.7　使用jQuery开发移动站点和应用程序

在为移动设备开发Web站点或应用程序时，我们可以像开发桌面Web站点或应用一样将jQuery集成进我们的项目。只要移动平台支持JavaScript，到现在为止我在前面各章提到的jQuery功能都能工作。

11.3　jQuery Mobile 预览版介绍

jQuery团队已经发布了jQuery Mobile插件（http://jquerymobile. com）的Alpha版本（参见图11-11）。有关它的讨论和小道消息不断在jQuery社区传播。会有一个单独的jQuery Mobile库吗？没错，它是一个大约6 KB大小的独立插件，要先包含jQuery库，再包含它方可使用。它将支持哪些平台？Apple iOS、Google Android、Blackberry、bada、Windows Phone、Symbian、Palm webOS，还有MeeGo。访问http://jqueymobile.com/gbs/可查阅完整的兼容性列表。它体积有多大？未压缩的开发版本约86 KB，最小化之后为12 KB。

因为jQuery Mobile现在还是Alpha版本[①]，bug或问题在所难免，但是从已经发布的支持级别及特性列表来看，它的前景相当不错。

---

① 翻译到这儿时BETA3已经发布。

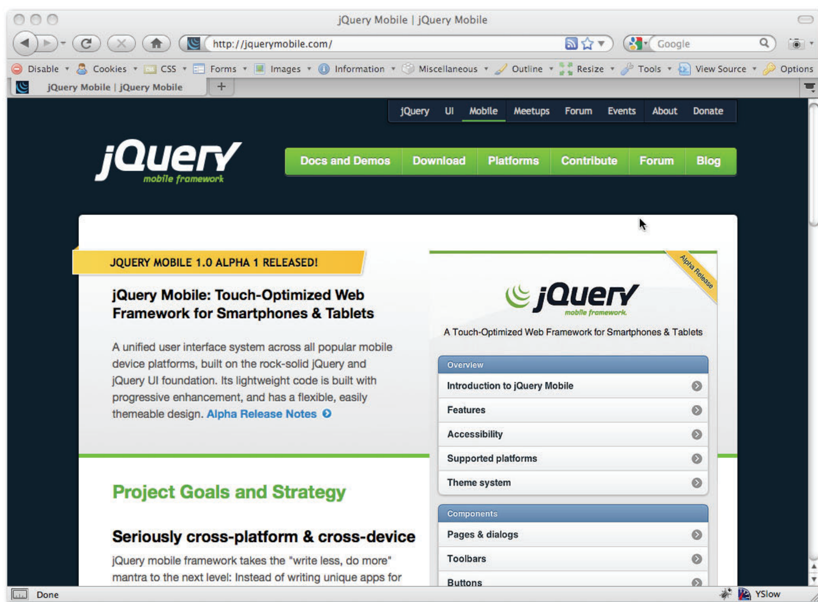


图11-11 jQuery Mobile首页

11.4 移动框架

移动框架与JavaScript库类似，它们提供一系列API，这些API负责与开发面向的移动设备通信。眼下jQuery只支持Appcelerator Titanium和jQTouch移动框架。

11.4.1 Appcelerator Titanium框架

Appcelerator发布于2008年12月，是一个用于创建移动和桌面应用的开发框架（参见图11-12）。使用这个框架，你可以使用HTML、CSS和JavaScript/jQuery技术，为Apple iPhone、Apple iPad、Google Android和黑莓手机开发移动应用。

Appcelerator Titanium Mobile支持以下特性：

- 让应用使用原生用户界面，这样用户就会对你的应用有似曾相识之感；
- 多媒体支持，在真实设备上支持照片、音乐和视频流；
- 支持存取本地文件，即支持在设备上访问文件和储存文件，一能提高应用程序性能，二能让应用可在无网络连接时使用；
- 访问移动设备的照相机和摄像头，以创建交互式应用；
- 支持地理定位，通过GPS创建使用地理位置信息的应用程序，进一步扩展用户体验；
- 支持HTML5和CSS3。

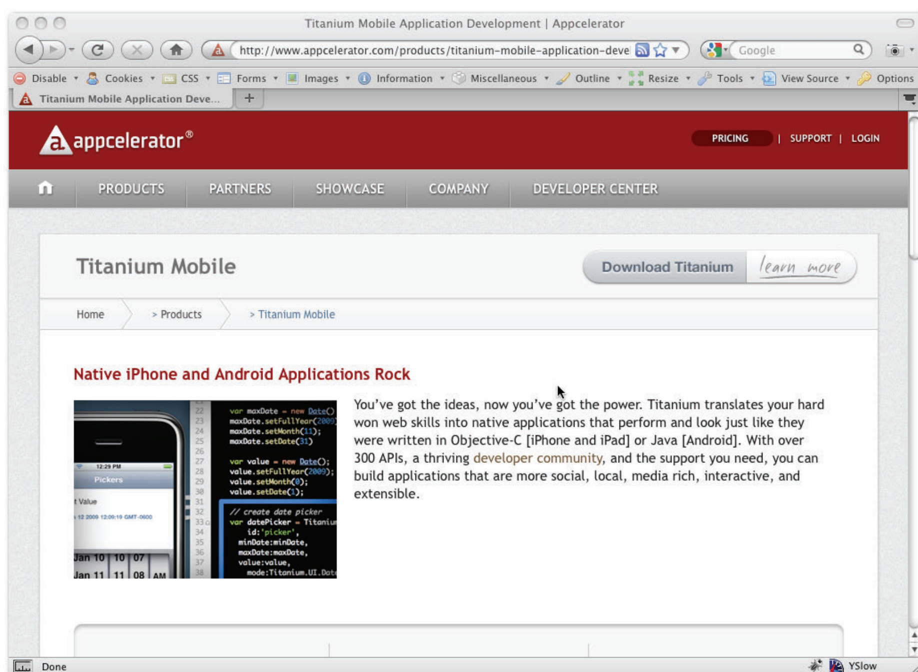


图11-12　Appcelerator Titanium Mobile官方网站

访问 www.appcelerator.com/products/titanium-mobile-application-development/可获取完整的功能列表。写到这里时（2010年秋天），Appcelerator 已经拥有4900个应用，且据称其支持社区拥有超过75 000名开发者。Appcelerator取得成功的原因之一是，它是一个开源项目，对那些缺乏坚实编程技术的Web设计人员来说，使用自己熟悉的技术创建应用程序真的是相当容易。

Appcelerator Titanium Mobile不是为那些纯粹的菜鸟准备的，但它的确提供了技术支持并有相当棒的在线文档，有了这些支援，新手就很容易起步。

11.4.2　jQTouch插件

jQTouch 是一个开源的jQuery插件，支持在为iPhone开发Web应用时模仿iPhone原生系统的一些功能，如动画、导航、主题等（参见图11-13）。jQTouch支持用HTML、CSS和jQuery实现这些功能。

最近jQTouch被Sencha收购，成为了Sencha Touch应用程序的一部分（参见图11-14）。Sencha Touch是一个基于HTML5的移动应用框架，能使我们为Apple iOS和Google Android创建Web应用。



图11-13　jQTouch官方网站



图11-14　Sencha Touch官方网站

第 12 章

jQuery资源

12

互联网上有太多专注于jQuery框架的官方/非官方站点和线上资源。jQuery官方站点提供了API查询、框架下载及bug追踪等功能，这些功能很有用，但官方网站提供的其他功能则不够出彩。与jQuery官方网站相比，jQuery社区则是一个无价的资源宝库，极其有效地弥补了官方网站的不足，提供了大量的教程、代码示例、插件等。本章，我们将一起回顾jQuery所有官方的和非官方的资源。

即便读完了这本书，我们仍应不断学习jQuery技术。jQuery社区是如此庞大，而且每天都在增长。如果用Google搜索jQuery，会看到1800万条结果，并且这个数字每时每刻都在增长。每个早晨，我都阅读Popurls（popurls.com），它聚合了像Digg（www.digg.com）、Delicious（www.delicious.com）、RedDit（www.reddit.com）、Twitter（www.twitter.com）这样的著名站点的优秀内容，还支持很多其他社交社区。几乎每一天，我都会读到至少一条有关jQuery某个功能的头条新闻。没错，jQuery是一个相当热的主题。

12.1 jQuery 的快速成长

jQuery的成长让人印象深刻。不久以前，它的用户还只有一小撮儿，现在连谷歌、微软这样的公司都在支持它，而且还有很多很多其他的公司（参见图12-1），支持jQuery的站点正在爆炸性地增长。甚至有些以前不打算支持jQuery的公司，现在已经完全投资在这个库上。

成功的jQuery由它的缔造者John Resig和他的团队，以及支持着jQuery的庞大社区共同打造。我鼓励这本书的每一位读者为jQuery项目捐赠。不必多大的金额，哪怕只有5美元，也会造就不同。jQuery项目是非赢利的。绝大多数为这个项目工作的人都是不领薪水的志愿者，他们的目标是打造一个极具价值的东西，而不是赚钱。

想想jQuery为我们节省的时间和脑力，捐赠几美元给他们是表达感谢的最好方式。为了让jQuery背后的开发者能更安心地打磨这一伟大的产品，欢迎到 http://jquery.org/donate 网页捐赠（参见图12-2）。

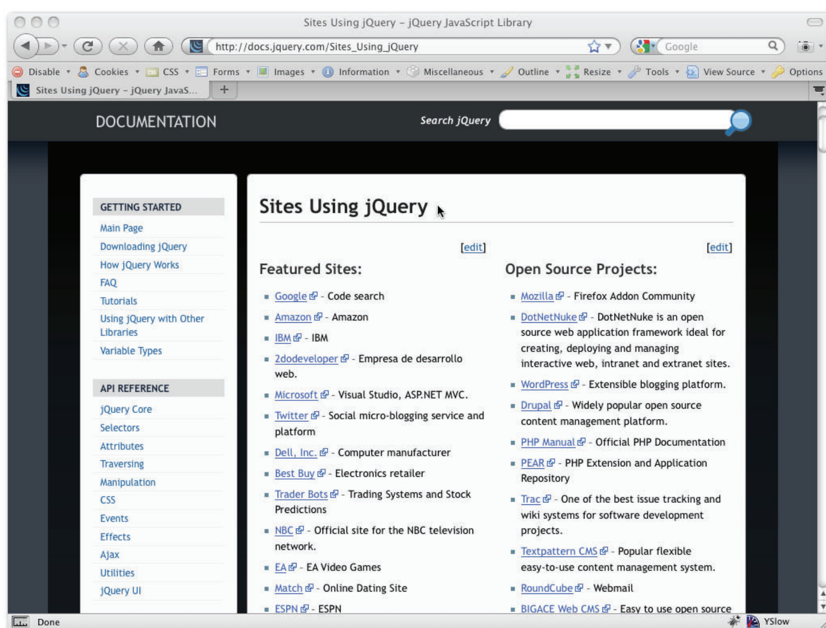


图12-1 使用jQuery的站点

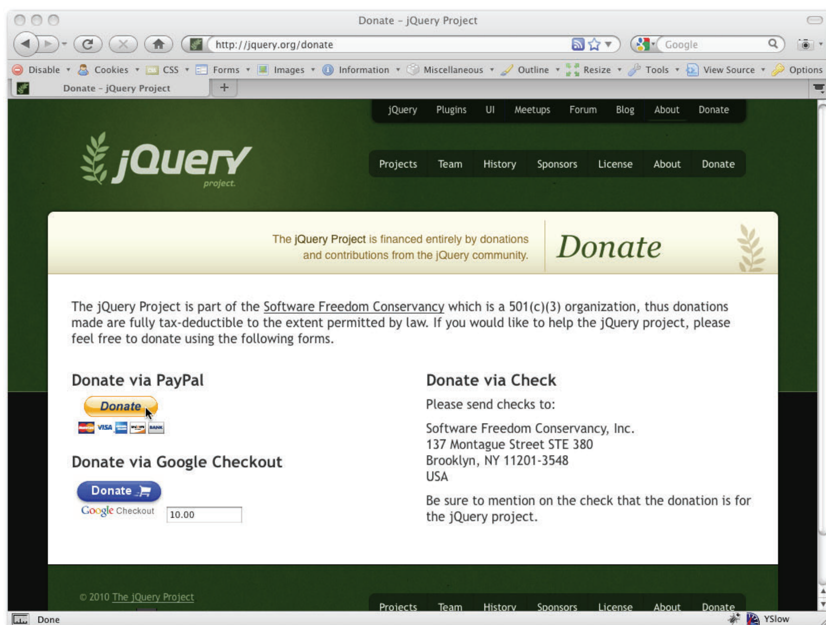


图12-2 jQuery项目捐赠页面

12.2　jQuery 官方站点

jQuery官方站点（jquery.com）提供了堪称完美的文档，介绍如何使用它的API。不过在其他方面，这个站点在易用性和一致性方面做得还不够。它由5部分组成：

- ❑ Download（下载）;
- ❑ Documentation（文档）;
- ❑ Tutorials（教程）;
- ❑ Bug Tracker（bug追踪）;
- ❑ Discussion（讨论）。

下载部分提供了下载jQuery库的多种方法，还介绍了通过CDN（内容分发网络）使用jQuery的方法。

12.2.1　jQuery API文档子站

jQuery幕后的团队花费了海量的时间打造这份完美到极致的在线文档，供任何人免费使用。它拥有当前版本及老版本中所有方法的详尽信息（参见图12-3）。API文档站点的网址是api.jquery.com。

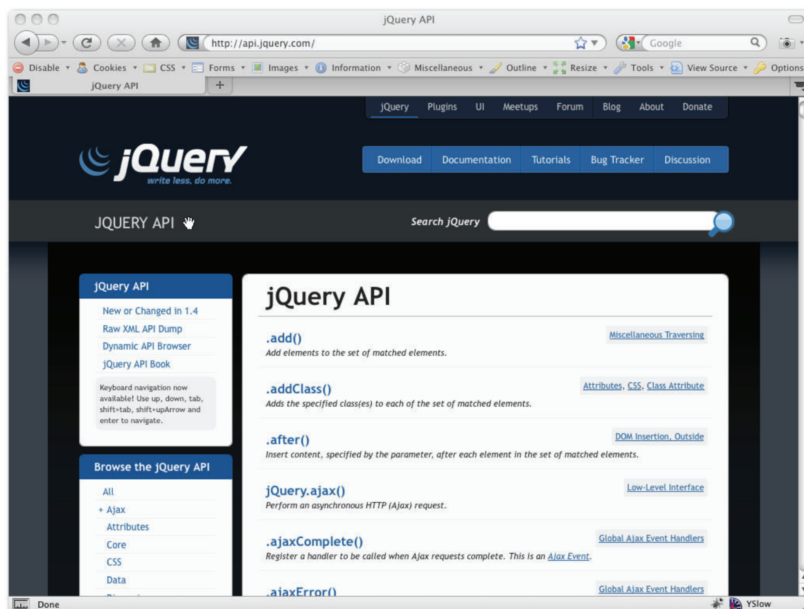


图12-3　jQuery API文档子站

jQuery团队还在jQuery站点上实现了相当好用的搜索功能，这让我们找起东西来更加容易。在这个站点上，我从没有在找东西方面遇到问题。搜索引擎也相当好地索引了jQuery站点，通过

搜索引擎一样能够方便地找到自己需要的API。

每一个函数或方法都有自己的专门页面，包括完善的描述、代码示例以及来自开发者的评论。图12-4展示的是css()方法的专属页面。

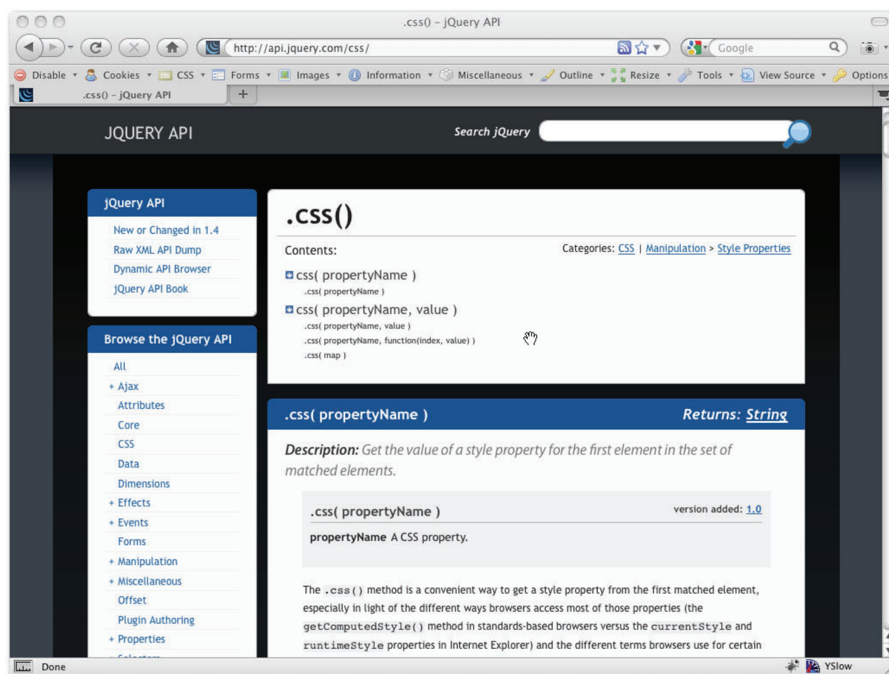


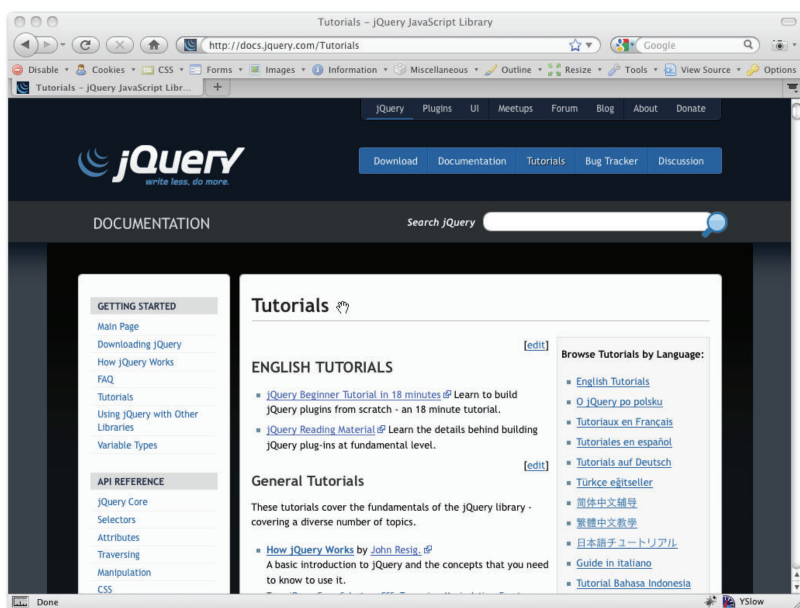
图12-4　API文档中的css()方法

12.2.2　jQuery教程

教程也是jQuery站点的一大服务，这些教程来自jQuery社区，由jQuery团队发表，见图12-5。这些教程涉及了API的方方面面，此外还有一些使用其他语言发表的教程。如果你想用jQuery做点新奇的事情，这是一个开始的好地方。jQuery教程服务的网址是docs.jquery.com/Tutorials。

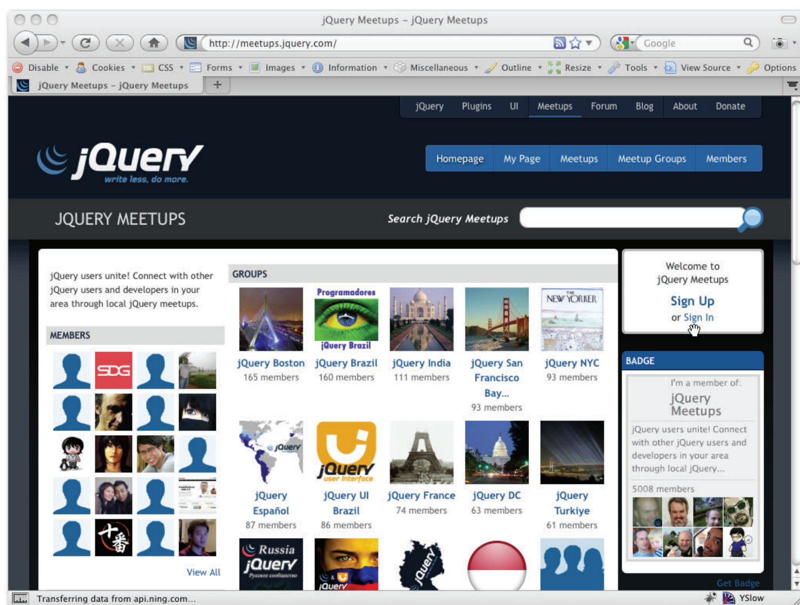
12.2.3　jQuery 聚会或讨论会

jQuery聚会的网址是http://meetups.jquery.com/。在发布1.3.2版本时，jQuery网站第一次在Meetup网站（www.meetup.com）的协助下提供jQuery聚会（Meetups）服务，见图12-6。这些聚会大都是由一些非官方的俱乐部或jQuery发烧友组织的，在这些聚会上大家互相认识并对jQuery进行更深入的了解。我强烈建议你寻找一个离自己比较近的jQuery聚会，参加一次。



© 2010，jQuery项目

图12-5　jQuery.com 解决方案



© 2010，jQuery 项目

图12-6　jQuery.com 聚会

除了jQuery社区组织的这些聚会，jQuery开发团队每年也会在旧金山或波士顿组织一到两次jQuery大会。jQuery大会吸引少而精的人群参会，人数一般为250 ~ 300，这些人大都是来自于jQuery核心团队或外部社区的、专注于jQuery有关议题的大演讲家。jQuery大会大大促进了那些专职开发jQuery的专家和新加入的开发者之间的交流。jQuery团队还在jQuery网站上专门为jQuery大会开辟了一角，介绍大会的见闻，见图12-7。还有一些别的有关JavaScript开发的专家会议，而jQuery大会是唯一专注于jQuery的大会。



图12-7　官方jQuery大会网站

12.2.4　bug追踪系统

在使用jQuery库的过程中，你可能会在编程过程中遇到jQuery的bug。jQuery团队创建了一个bug追踪系统（参见图12-8），jQuery用户可以提交bug到这个系统，这样他们就能在将来的版本中修正。追踪bug很重要，而jQuery有着如此大的用户社区，使用bug追踪系统有组织地提交bug给jQuery开发团队当然是非常必要的。jQuery的bug追踪系统网址是http://bugs.jquery.com/newticket?redirectedfrom=。

12.2.5　jQuery论坛

jQuery团队还架设了一个论坛（参见图12-9），在那里可以发表任何有关jQuery的东西，包括赞扬的话、问题、教程、代码示例等。这里是各种技术水平的jQuery开发者相互交流的好地方。

本书写作时，jQuery论坛已经有65 000多个主贴和143 236篇回复。这是一个相当活跃的社区，

只需免费注册一个账号就可参与其中。jQuery论坛的网址是http://forum.jquery.com/。

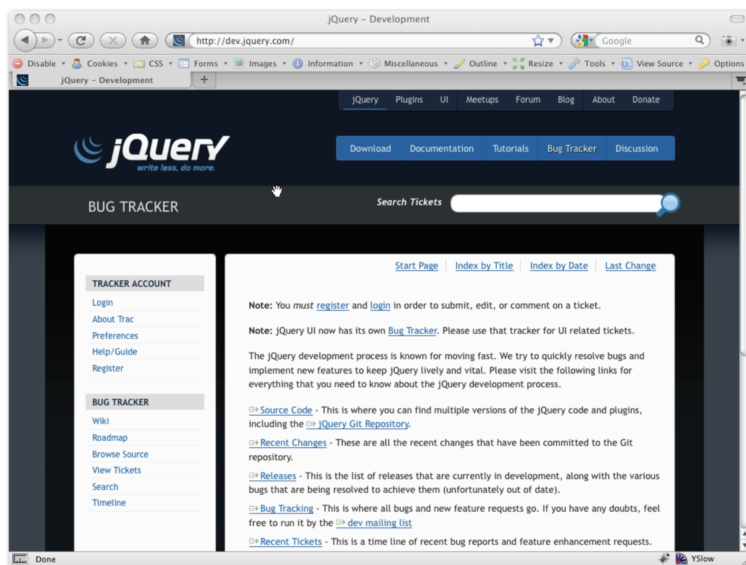


图12-8　jQuery.com 的bug追踪系统

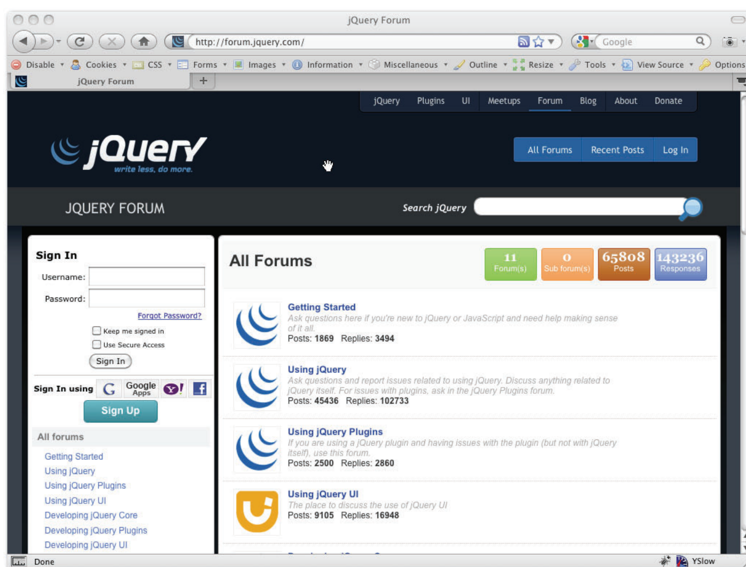


图12-9　jQuery.com论坛

12.3　其他 Web 设计和开发资源

下面是一些对开发者有帮助的有关jQuery和其他Web设计主题的优秀站点：

- Learning jQuery（www.learningjquery.com）;
- Stack Overflow（www.stackoverflow.com）;
- jQuery for Designers（www.jqueryfordesigners.com）;
- Visual jQuery（www.visualjquery.com）;
- The 14 Days of jQuery（www.jquery14.com）;
- Nettuts+（www.nettuts.com）;
- Ajaxian（www.ajaxian.com）;
- Forrst（www.forrst.com）;
- SitePoint（www.sitepoint.com）。

通过创建更具交互性的富Web界面增强用户体验

Smashing jQuery

精彩绝伦的jQuery

本书阐述如何利用少量的JavaScript基础知识将jQuery框架整合进网站，以创建富Web界面，并建立兼容所有主流浏览器的交互性网站。作者是一位具有丰富经验的Web设计师和开发者，通过一系列指导性步骤清晰讲述了添加交互性以创建卓越Web应用的方法和技巧。

使用jQuery可以节省大量的开发时间，开发者在没有丰富编程经验的情况下也能编写出超乎想象的交互组件。本书包含大量实用技巧、解决方案和案例，可极大地促进Web应用的开发与设计。

书中主要知识点包括jQuery基础知识、Ajax请求、事件与效果、DOM操作（包含构建下拉菜单等任务的教程）、lightbox窗口、表单管理、动态的表格数据、鼠标事件效果、模态对话框、自定义jQuery插件等。

Jake Rutter　Web设计师和开发者，有9年多的用户界面设计和前端开发经验，精通HTML、CSS和JavaScript。

- Smashing杂志专业策划
- 阐释jQuery的现代技巧和最佳实践
- 初学者入门的捷径



WILEY
www.wiley.com

图灵社区：www.ituring.com.cn
新浪微博：@图灵教育　@图灵社区
反馈/投稿/推荐信箱：contact@turingbook.com
热线：(010)51095186转604

分类建议　计算机/网络技术/jQuery

人民邮电出版社网址：www.ptpress.com.cn



ISBN 978-7-115-28065-7
定价：59.00元